VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Porovnání běhů algoritmu diferenciální evoluce pomocí sítí

Comparisons of the Differential Evolution Algorithm Runs by Means of Networks

2018 Jakub Komoráš

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Jakub Komoráš**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Porovnání běhů algoritmu diferenciální evoluce pomocí sítí
Comparisons of the Differential Evolution Algorithm Runs by Means of Networks

Jazyk vypracování: čeština

Zásady pro vypracování:

Biologicky inspirované algoritmy jsou založeny na nejrůznějších jevech v přírodě. Tyto algoritmy nám umožňují nalézt řešení problémů, které jsou klasickými metodami velmi obtížné či dokonce neřešitelné. Příkladem takovýchto algoritmů mohou být neuronové sítě, evoluční algoritmy, kolektivní inteligence či jiné. Pro pochopení jejich dynamiky můžeme využít například sítě (komplexní sítě). S pomocí sítí pak můžeme algoritmy analyzovat a následně algoritmy vylepšit.

Úkolem této práce je prostudovat oblast biologicky inspirovaných algoritmů, hlavně pak algoritmu diferenciální evoluce, a teorie sítí, navrhnout převody tohoto algoritmu na síť a následně porovnat získané sítě pro různé běhy.

Cíle této práce je možné shrnout v těchto bodech:
1. Nastudovat a popsat problematiku biologicky inspirovaných výpočtů.
2. Nastudovat a popsat algoritmus diferenciální evoluce.
3. Nastudovat a popsat problematiku sítí (komplexních sítí).
4. Naprogramovat algoritmus diferenciální evoluce s různými strategiemi.
5. Navrhnout a naprogramovat převody daného algoritmu na síť.
6. Program vhodně otestovat.
7. Porovnat vlastnosti získaných sítí pro různé běhy daného algoritmu - různé parametry, různé účelové funkce, lepší a horší nalezené řešení.

Seznam doporučené odborné literatury:

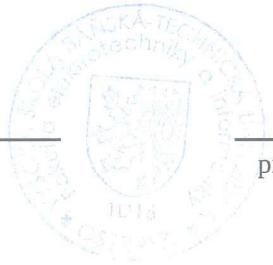
[1] BOCCALETTI, Stefano, et al. Complex networks: Structure and dynamics. Physics reports, 2006, 424.4: 175-308.
[2] DAS, Swagatam; MULLICK, Sankha Subhra; SUGANTHAN, Ponnuthurai N. Recent advances in differential evolution–an updated survey. Swarm and Evolutionary Computation, 2016, 27: 1-30.
[3] NEWMAN, Mark. Networks: an introduction. 2010. United Slates: Oxford University Press Inc., New York
[4] STORN, Rainer; PRICE, Kenneth. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization, 1997, 11.4: 341-359.
[5] ZELINKA, Ivan, et al. Evoluční výpočetní techniky: principy a aplikace. 1. vyd. Praha: BEN, 2009, 534 s. ISBN 978-80-7300-218-3.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Lukáš Tomaszek**

Datum zadání: 01.09.2017

Datum odevzdání: 30.04.2018



doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 30. dubna 2018 .....

Rád bych na tomto místě poděkoval všem lidem, kteří mi byli nápomocni při tvorbě této práce. Převážně svému vedoucímu panu Ing. Lukáši Tomaszkovi za pravidelné konzultace, zodpovědné vedení a věcné připomínky.

Abstrakt

Tato bakalářská práce se zabývá zaznamenáváním průběhů algoritmu diferenciální evoluce, jejich převodem na síť a následným porovnáním mezi lepšími a horšími průběhy. K tomuto účelu byly v rámci této práce implementované dva nástroje. Jsou zde nastíněny principy fungování biologicky inspirovaných výpočtů a evolučních algoritmů s detailním zaměřením na algoritmus diferenciální evoluce. V další kapitole jsou definovány pojmy jako orientované a vážené sítě, které jsou potřebné pro následnou analýzu. Ještě před samotným zhodnocením dat jsou zde popsány použité nástroje. V poslední kapitole se práce zaměřuje na analýzu vygenerovaných dat a hledání rozdílů mezi lepšími a horšími běhy algoritmu DE.

Klíčová slova: biologicky inspirované výpočty, evoluční algoritmy , diferenciální evoluce

Abstract

This bachelor thesis deals with the recording the runs of the algorithm differential evolution. Next, runs are converted to the networks and a comparison is made between the better and the worst ones. For this purpose, two tools have been implemented in this thesis. Principles of biologically inspired computations and evolutionary algorithms with a detailed focus on algorithm differential evolution are described. The following chapter defines the concepts of oriented and weighted networks that are needed for subsequent analysis. Before the data evaluation, the tools used here are described. In the last chapter, the thesis focuses on the analysis of generated data and the search for differences between better and worse runs of differential evolution algorithm.

Key Words: bio-inspired computing, evolutionary algorithms, differential evolution

Obsah

Seznam použitých zkratek a symbolů

BC	–	Betweenness centralita
BIA	–	Biologicky inspirované algoritmy
BIV	–	Biologicky inspirované výpočty
CC	–	Closeness centralita
DE	–	Diferenciální evoluce
EC	–	Eigenvector centralita
GCC	–	Globální koeficient shlukování
LCC	–	Koeficient shlukování pro jednotlivý vrchol
WCC	–	Vážený koeficient shlukování

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

1 Úvod

Biologicky inspirované výpočty jsou oblastí dostávající se v poslední době do popředí [15]. Patří do skupiny heuristických metod, které jsou nasazovány na řešení problémů, kde výpočty pomocí exaktních metod selhávají v časové náročnosti. Jinými slovy jsou pomocí těchto exaktních metod tyto výpočty neproveditelné v limitovaném časovém období. BIV jsou algoritmy inspirované přírodou. Tyto algoritmy se snaží například o využití kolektivního chování či evoluce.

Součástí BIV jsou například genetické algoritmy [11] využívané k řešení problémů s maximalizací / minimalizací jako je problém obchodního cestujícího [13]. Za dalšího zástupce BIV lze považovat optimalizaci pomocí mravenčích kolonií, využívající principů rojové inteligence, která byla kupříkladu využita na řešení problému návrhu jednoduché výrobní linky, tak aby došlo k minimalizaci počtu stanovišť výroby [19]. Detailnějším popisem BIV se v této práci zabývá první kapitola.

V návaznosti na BIV se druhá kapitola zabývá evolučními algoritmy. Tyto algoritmy využívají principy evoluce, kdy hlavním prvkem je předávání rodičovského genomu potomkům, postupný vývoj a přežití jen nejsilnějších jedinců [1]. V této kapitole je rozepsán princip fungování těchto algoritmů. Zajímavostí je, že průběh evolučních algoritmů je z jisté části řízen náhodou, což způsobuje, že výsledek není možné dopředu přesně předpovídat. Z důvodu využívání náhody se k tomuto druhu algoritmů velmi obtížně sestavují matematické důkazy, proto se při posuzování použitelnosti těchto algoritmu vychází z praktických zkušeností. Evoluční algoritmy byly například využity při optimalizaci spotřeby paliva u motorů firmy Boeing pro letecký průmysl. Díky této optimalizaci došlo k velmi citelnému ušetření nákladů [39]. Jedním z evolučních algoritmů je algoritmus diferenciální evoluce [35].

Ve třetí kapitole této práce je popsán vybraný evoluční algoritmus, konkrétně algoritmus DE. Algoritmus DE je oproti ostatním evolučním algoritmům specifický v jedné věci, a to té, že k vytváření nových potomků je využíváno většího počtu rodičů než pouhých dvou [39]. Tento algoritmus je velmi často využíván při optimalizaci. Optimalizací rozumíme snahu o maximalizování požadovaných vlastností systému a současně potlačení vlastností nežádoucích. Za tímto účelem byl algoritmus DE využit i při optimalizaci výroby radiálních aktivních magnetických ložisek tak, aby došlo k maximalizaci nosné síly ložiska a současně ke snížení jeho hmotnosti [28].

Další kapitola se zabývá problematikou sítí. Pod pojmem síť si lze představit jednoduchou strukturu sestavenou z vrcholů a jejich propojení neboli hran. Pokud se nad tímto zamyslíme, jsou všude kolem nás. Každá osoba je součástí sítě z pohledu sociálních vztahů, kdy jednotlivé osoby představují vrcholy a hrany nesou informace o sociálních interakcích, jako například přátelství. Za další příklady sítí v reálném životě můžeme považovat elektrické rozvodné sítě [25], internet [21], sociální [31] a silniční sítě [8].

Studiem sítí se zabývá větev diskrétní matematiky nazývaná teorie grafů [4]. Základy této disciplíny položil Leonhard Euler, kdy v osmnáctém století vyřešil situaci známou jako problém

sedmi mostů města Královce. Z pohledu sítí tento problém vyřešil za pomocí hledání cesty tak, aby každou hranu obsahovala pouze jednou. Tímto se dostáváme k praktickému využití této disciplíny, kdy za pomoci sítí lze například výhodně plánovat trasy popelářských vozů a poštovních doručovatelů za účelem nalezení nejefektivnější trasy [17].

Šestá a sedmá kapitola popisuje dva nástroje použité při následné analýze. První z nástrojů byl implementován pro účel generování záznamů z průběhu algoritmu DE, kde výsledkem jsou textové soubory nesoucí informace o vývoji jedinců v každé generaci. Druhý z nástrojů navazuje na tyto soubory a jednotlivé záznamy běhů převádí pomocí několika metod na sítě. Tyto sítě je následně možné tímto nástrojem vizualizovat. Další z možností využití tohoto nástroje je výpočet vlastností zmíněných ve čtvrté kapitole.

V poslední kapitole je za využití dvou implementovaných nástrojů nastíněna analýza a vyhodnocení. Tato analýza je předvedena na jedné verzi algoritmu DE pro implementované typy převodů na síť a pro čtyři různé účelové funkce. Každá účelová funkce byla zvolena z jiné kategorie tak, aby byly pokryté všechny kategorie vyskytující se v rámci testovacích funkcí.

Cílem této bakalářské práce je porovnání dvou skupin běhů algoritmu DE. Do první skupiny jsou zařazeny běhy, které dosáhly nejlepších výsledků na dané účelové funkci. Zatím co v druhé skupině jsou běhy s výsledky nejhoršími. Po následném převodu na síť budou zkoumány rozdíly mezi těmito skupinami pro jednotlivé vlastnosti. Toto porovnání by mělo pomoci odhalit rozdíly mezi dobrým a špatným průběhem algoritmu DE tak, aby bylo možné na tomto základě upravit řídící parametry a snažit se o zvýšení úspěšnosti při využití algoritmu DE.

Z důvodu čerpání informací převážně ze zahraniční literatury se v práci můžou vyskytovat některé pojmy v anglickém znění, jelikož pro ně neexistuje rozumný překlad. Toho jevu jsem si vědom a vyskytuje se například při názvech testovacích funkcí, kde byly ponechány jejich anglické názvy.

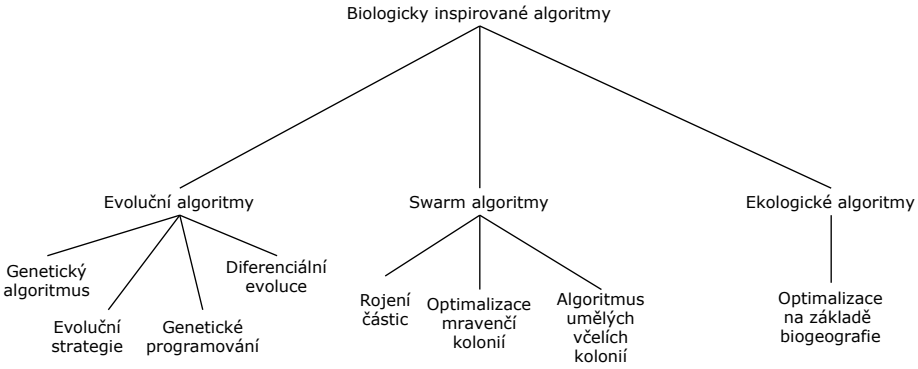
2 Biologicky inspirované výpočty

Veškeré biologické systémy profitují z evolučního procesu, tedy postupného vývoje. Schopnost vývinu a adaptace u biologických systémů v přírodě dodala motivaci ke snaze o promítnutí těchto principů do softwarových a hardwarových řešení [10]. Algoritmy inspirované přírodou mají schopnost popsat a vyřešit složité vztahy z velmi jednoduchých počátečních podmínek a pravidel s pouze malou nebo žádnou znalostí prohledávaného prostoru. BIV přichází jako nová éra výpočetní techniky zahrnující širokou škálu aplikací, zahrnující skoro všechny oblasti včetně počítačových sítí, bezpečnosti, robotiky, zpracování dat a mnoho dalších [3].

Klasické možnosti řešení problémů můžeme rozdělit do dvou kategorií, dle způsobu použitých metod. Exaktní metody, využívající logiku a matematické programování, nebo heuristické metody, do kterých spadají právě BIV. Heuristické metody nacházejí své uplatnění v řešení těžkých a komplexních optimalizačních problémů, na kterých tradiční metody selhávají [15].

Zástupci BIV se nazývají biologicky inspirované algoritmy. BIA se snaží imitovat strategii procesů v přírodě, protože mnoho biologických procesů lze považovat za procesy optimalizace. BIA mohou využívat častá náhodná rozhodnutí. Aby bylo možné problém řešit pomocí BIA musí být správně zvolena jeho reprezentace. Vyhodnocení jednotlivých nalezených výsledků probíhá pomocí tzv. účelové funkce. Pro tuto funkci musí být také správně definované operátory, využívané k vzájemnému porovnávání výsledků [3].

Hlavní dva proudy BIA jsou reprezentovány evolučními a swarm algoritmy, které jsou inspirovány evolucí v přírodě a chováním kolektivu [3]. Kategorie spadající pod BIA jsou znázorněny na obrázku 1 včetně zástupců z jednotlivých skupin.



Obrázek 1: Členění BIA

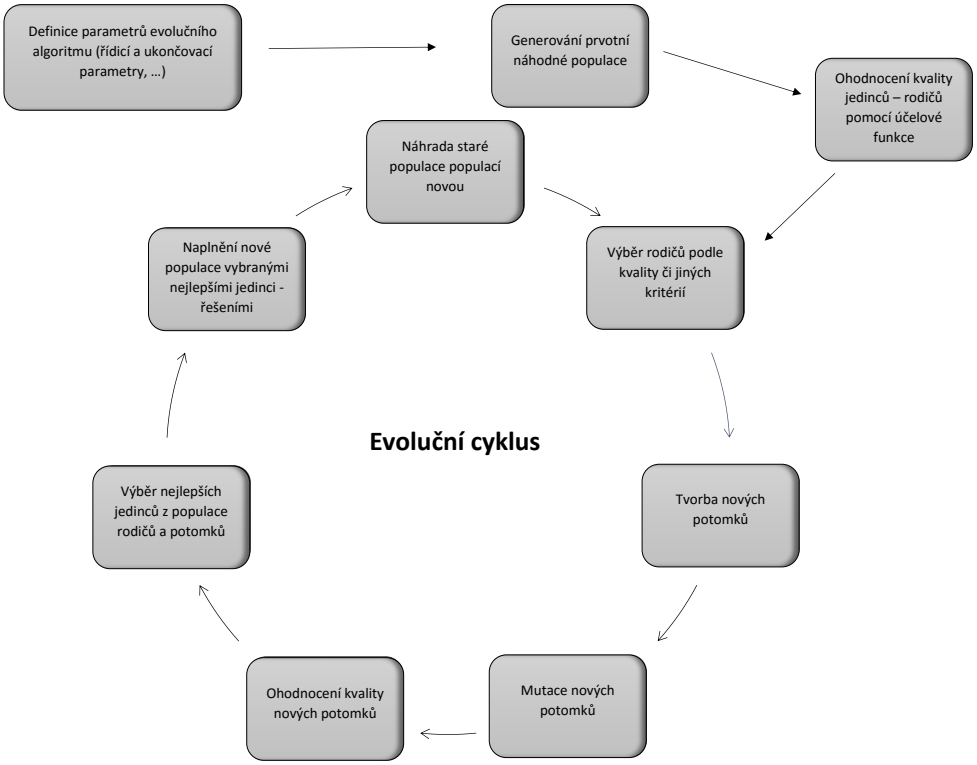
Swarm inteligence neboli rojová inteligence se zabývá komplexními přírodními systémy na základě kolektivního chování. Jednotliví jedinci roje nejsou nikým řízeni a každý tento jedinec se chová jen dle svého vnímání okolí [16].

U evolučních algoritmů hraje hlavní roli předávání rodičovského genomu nově vzniklým potomkům [39]. Tato kategorie BIA je detailněji rozebrána v následující kapitole.

3 Evoluční algoritmy

Evoluční algoritmy vycházejí z principů evoluce, kdy dochází k předávání rodičovského genomu potomkům a kde následně rodičovský genom zaniká. Tyto algoritmy byly postaveny na základě Darwinovy a Mendelovy teorie evoluce. V této teorii je zmíněno evoluční dogma, dle kterého dochází k vývoji jednotlivých druhů na základě plození potomků rodiči a potomci podléhají během svého vzniku mutacím. Jedinci nejhůře přizpůsobení prostředí následně v cyklech, neboli generacích, zanikají, čímž dochází k uvolnění místa pro nové a lepší jedince.

Evoluční algoritmy patří do skupiny evolučních výpočetních technik. Další příklady evolučních výpočetních technik jsou například genetické programování [18] a evoluční hardware [39]. Na obrázku 2 je schématicky znázorněn průběh evolučního algoritmu. V průběhu algoritmu dochází v prvním bodě k definici a vymezení parametrů pro řízení běhu a ukončení, stanovení účelové funkce pro vyhodnocování jedinců a řešení daného problému.



Obrázek 2: Obecné schéma evolučního algoritmu

Následuje krok ve kterém dochází ke generování prvotní populace, která se skládá z určeného počtu jedinců a náhodném generování parametrů těchto jedinců. Každý jedinec populace představuje právě jedno řešení účelové funkce. Každý jedinec z populace je poté ohodnocen účelovou funkcí. Tato hodnota je buď přímo, nebo po normalizaci, přidělena ke každému jedinci a slouží pro porovnání vůči ostatním jedincům. Po provedení těchto kroků dochází k zahájení tzv. evo-

lučního cyklu, který se neustále opakuje, dokud není splněna jedna z podmínek pro ukončení běhu, či není splněno kritérium na kvalitu řešení.

Jako první je nutné v evolučním cyklu zvolit rodiče, kteří budou použiti k tvorbě potomků. Tato selekce probíhá například podle jejich hodnoty účelové funkce, nebo také náhodně. Ze zvolených rodičů křížením vznikají potomci. Průběh tohoto procesu se liší pro jednotlivé algoritmy. Po vytvoření potomka dochází k jeho mutaci což znamená, že dochází k pozměnění parametrů jedince za pomoci náhodných procesů, tak aby se proces vytváření jedince co nejblíže přiblížil biologickým mutacím genů v přírodě. Jakmile je jedinec vytvořen dochází k jeho ohodnocení účelovou funkcí, aby bylo možné uskutečnit jeho porovnání s ostatními a vyberou se nejlepší z nich. Tímto dochází k vytvoření nové populace, dochází k zániku populace původní a začíná nový evoluční cyklus, tentokrát pro nově vytvořenou generaci.

Kategorie evolučních algoritmů, je tvořena mnoha různými algoritmy, které mají podobný průběh, znázorněný obecně na obrázku 2. Hlavními rozdíly jsou odlišné procesy probíhající v rámci křížení a mutace. Například u některých algoritmů probíhá křížení rodičů přehozením jejich částí, v jiných se tvorba potomka podobá vektorovým operacím [39]. Zde jsou příklady algoritmů spadající do kategorie evoluční:

- genetické algoritmy [12],
- evoluční strategie [30],
- genetické programování [18],
- diferenciální evoluce [35].

Algoritmus diferenciální evoluce je podrobně rozepsán v následující kapitole.

4 Diferenciální evoluce

Jedná se o jeden z evolučních algoritmů. Tento algoritmus byl poprvé použit v roce 1995. Vyvinuli jej Ken Price a Rainer Storm [34]. První verze DE vycházela z algoritmu genetické žíhání [38]. Jednalo se o změnu genetického žíhání z binární reprezentace do reprezentace dekadické, společně s tím byly změněny operace logické na vektorové. Po těchto změnách přišel Ken Price s tzv. diferenciální mutací. Diferenciální mutace spočívá v generaci zkušebních řešení přičtením rozdílu dvou náhodně zvolených jedinců ke třetímu jedinci. Dále byly použity také metody selekce z genetického žíhání.

Později byly principy žíhání aplikované v tomto algoritmu odebrány. Zároveň Storn navrhl tvorbu nových potomků ve speciální populaci. Toto byla druhá verze DE. V roce 1996 se Storn a Price rozhodli algoritmus publikovat. Dosavadní verze však byly nedostačující pro řešení široké množiny optimalizačních problémů, a proto vyvinuli třetí verzi [39].

4.1 Postup činnosti diferenciální evoluce

Diferenciální evoluci můžeme rozdělit do čtyř základních kroků dle pořadí, v jakém probíhají.

1. Stanovení parametrů
2. Tvorba populace
3. Evoluční cyklus
4. Testování naplnění ukončovacích parametrů

Tyto kroky si nyní podrobně rozebereme.

4.1.1 Stanovení parametrů

V tomto kroku dochází k nastavení parametrů, které řídí chod celé evoluce. Jedná se o mutační konstantu (F), práh křížení (CR), počet jedinců v populaci (NP) a dimenzi problému (D). Dále je v tomto kroku nutné nadefinovat vzorového jedince. Doporučené hodnoty řídících parametrů jsou uvedeny v tabulce 1.

CR - Práh křížení

Tento parametr řídí, jak moc se dokáže prosadit genom rodiče při tvorbě potomka. Čím nižší hodnota, tím více genomu z aktuálního rodiče potomek získá a naopak čím větší hodnota, tím více získá potomek genomu ze zbylých rodičů. Může nabývat hodnot od nuly do jedné, těmto krajním hodnotám je však doporučené se vyhnout. V případě nastavení hodnoty CR rovné nule dojde k tomu, že zkušební jedinec bude kopií aktuálního rodiče. V opačném případě bude zkušební jedinec tvořen pouze ze tří náhodně vybraných rodičů, bez zohlednění jejich kvality.

D - Dimenze problému

Značí počet argumentů účelové funkce. Tento parametr je dán řešeným problémem, lze ho změnit pouze, pokud změníme formulaci problému.

NP - Počet jedinců v populaci

Udává velikost populace. Doporučené rozmezí je od 10D do 100D. Hodnota NP by neměla být menší než je počet jedinců potřebných k evolučnímu cyklu, jelikož se rovná o minimální populaci, při které DE funguje.

F - Mutační konstanta

Využívá se při tvorbě nového potomka. Mutační konstantou je násoben diferenční vektor a tím dojde k jeho mutaci. Její hodnota se pohybuje v mezích od nuly do dvou.

Počet generací

Poslední parametr volí počet evolučních cyklů, neboli generací.

Tabulka 1: Hodnoty řídicích parametrů DE [39]

Řídící parametr	Interval	Doporučená hodnota
NP	[10D, 100D]	[10D]
F	[0, 2]	0,3 - 0,9
CR	[0 ,1]	0,8-0,9

4.1.2 Tvorba populace

Populace je typickým rysem všech evolučních algoritmů. Populaci si můžeme představit jako matici o rozměrech $N \times M$, kde N jsou sloupce představující jedince a M jsou řádky přestavující hodnoty parametrů jedince. Každému jedinci je také přiřazena hodnota fitness - vhodnost. Tato hodnota určuje jak moc je jedinec vhodný pro další populační vývoj. Hlavní činností všech evolučních algoritmů je neustálé vytváření nových populací, které postupně nahrazují staré a celá populace se tímto vyvíjí.

Pro vytvoření populace je potřeba definovat vzorového jedince, tzv. specimen. Tento jedinec slouží k vygenerování počáteční populace, a také k opravě parametrů jedinců, kteří překročí prohledávaný prostor. Vzorový jedinec obsahuje pro každý parametr typ proměnné a interval, v jakém se mohou pohybovat hodnoty daného parametru. Interval hodnot je velmi důležitý aspekt, eliminujeme pomocí něj například řešení, které by nebylo možné realizovat. Počáteční populace se generuje dle rovnice (1).

$$x_{i,j}^{(0)} = rnd[0,1] * (x_{i,j}^{(Hi)} - x_{i,j}^{(Lo)}) + x_{i,j}^{(Lo)} \tag{1}$$

Kde $x_{i,j}^{(0)}$ značí j-tý parametr i-tého jedince z počáteční populace. $x_{i,j}^{(Hi)}$ je maximální hodnota j-tého parametru definována vzorovým jedincem a $x_{i,j}^{(Lo)}$ je minimální hodnota j-tého parametru definována vzorovým jedincem.

4.1.3 Evoluční cyklus

Tento cyklus je postupně prováděn z pohledu každého jedince v populaci. Průběh evolučního cyklu lze rozdělit do tří částí: mutace, křížení, selekce [33]. DE má oproti jiným evolučním algoritmům tu zvláštnost, že k vytvoření potomka je potřeba čtyř nebo více rodičů, v závislosti na použité verzi, místo pouhých dvou.

V první části probíhá mutace. K aktivnímu jedinci (vektoru) se zvolí další jedinci z populace v počtu dle použité verze DE, například pokud je zvolena verze DE/rand/1 jsou to jedinci tři. Dále je popsána tvorba šumového vektoru pro tuto zvolenou verzi. První dva od sebe odečteme a získáme diferenční vektor. Tento vektor násobíme mutační konstantou F, čímž dostaneme zmutovaného jedince nazývaného váhový diferenční vektor. Váhový diferenční vektor následně přičteme ke třetímu vybranému jedinci a vznikne šumový vektor. Šumový vektor je mutací kombinace těchto tří jedinců. Mutace je prováděna dle rovnice (2). Poté následuje křížení.

U křížení se opět nachází jedna zvláštnost, a to že křížení u DE probíhá až po mutaci. Hlavním úkolem křížení je sestavení tzv. zkušebního vektoru, ten vznikne kombinací aktivního jedince a šumového vektoru. Princip jakým zkušební vektor vzniká závisí od typu použité mutace. Dva nejčastěji používané způsoby tvorby zkušebního vektoru jsou popsány níže.

Posledním krokem evolučního cyklu je selekce. Na pozici cílového jedince v nové populaci se vybere mezi aktivním jedincem a zkušebním vektorem ten, který má lepší hodnotu fitness. Proces evolučního cyklu je zobrazen na obrázku 3.

4.1.4 Testování naplnění ukončovacích parametrů

Ukončení běhu DE je nejčastěji po provedení určitého počtu cyklů definovaných uživatelem jako počet generací. Lze samozřejmě použít i jiné ukončovací podmínky, jako například počet ohodnocení jedinců, či zastavení průběhu algoritmu, když se hodnota fitness nejlepšího jedince nezmění po daný počet evolučních cyklů.

4.2 Verze algoritmu DE

Jednotlivé verze jsou značeny ve tvaru DE/x/y/z, kde DE značí algoritmus diferenciální evoluce, x reprezentuje řetězec označující vektor, který má být perturbován, a y je počet vektorů využitých pro perturbaci vektoru x, z zde představuje typ mutace, například exp pro exponenciální a bin pro binomickou. Zde vidíme pět nejčastěji využívaných mutačních strategií, pro tvorbu šumového vektoru [7] :

- DE/rand/1

$$v_j = x_{r1,j}^G + F(x_{r2,j}^G - x_{r3,j}^G) \tag{2}$$

- DE/best/1

$$v_j = x^G_{best,j} + F(x^G_{r1,j} - x^G_{r2,j}) \tag{3}$$

- DE/current-to-best/1

$$v_j = x^G_{i,j} + F(x^G_{best,j} - x^G_{i,j}) + F(x^G_{r1,j} - x^G_{r2,j}) \tag{4}$$

- DE/best/2

$$v_j = x^G_{best,j} + F(x^G_{r1,j} - x^G_{r2,j}) + F(x^G_{r3,j} - x^G_{r4,j}) \tag{5}$$

- DE/rand/2

$$v_j = x^G_{r1,j} + F(x^G_{r2,j} - x^G_{r3,j}) + F(x^G_{r4,j} - x^G_{r5,j}) \tag{6}$$

Kde v_j je j-tý parametr šumového vektoru a $x^G_{rN,j}$ znamená j-tý parametr N-tého náhodně vybraného jedince. Indexy r1, r2, r3, r4 a r5 jsou celá čísla náhodně vybraná z rozsahu [1, NP], generovaná při tvorbě každého šumového vektoru, a zároveň odlišná od indexu i, který je použit pro označení aktivního jedince. F značí mutační konstantu. x^G_{best} představuje jedince s nejlepším fitness v populaci pro generaci G.

4.3 Typy mutací

U DE se mohou vyskytovat různé typy mutací, které jsou definovány v označení DE/x/y/z jako z. U DE se nejběžněji používají dva typy mutace a to binomická a exponenciální [7]. Oba tyto typy jsou závislé na řídicím parametru CR.

První z nich, tedy binomická mutace, spočívá ve vygenerování náhodného čísla v rozmezí [0, 1]. Pokud je toto číslo menší než konstanta CR, bude do zkušebního vektoru doplněna hodnota ze šumového vektoru. Tento princip vyjadřuje rovnice 7, kde K je náhodně zvolené číslo v rozmezí [1, D] a zajišťuje, aby alespoň jedna komponenta ve zkušebním vektoru byla z vektoru šumového. v značí šumový vektor a x aktivního jedince. Náhodné číslo, pro porovnávání s parametrem CR, je generováno pro každou komponentu.

$$u_{i,j} = \begin{cases} v_{i,j} & pokud\ j = K\ nebo\ rand_{i,j} \leq CR \\ x_{i,j} & jinak \end{cases} \tag{7}$$

Dalším typem je exponenciální mutace. U tohoto typu dochází v první řadě ke zvolení náhodného čísla n z rozmezí [1, D]. Toto číslo je použito jako startovací bod, ze kterého začíná výměna komponent. Dále se zvolí další náhodné číslo L z rozsahu [1, D], které určuje kolik komponent bude ze šumového vektoru obsaženo ve vektoru zkušebním. Volba čísla L probíhá dle následujícího pseudokódu:

```
L = 0;
DO
{
     L=L+1;
}WHILE ((rand[0,1]<CR)AND(L<D));
```

Výpis 1: Volba čísla L pro počet komponent ze šumového vektoru

Jakmile dojde ke zvolení čísel n a L, je zkušební vektor vytvořen dle rovnice 8

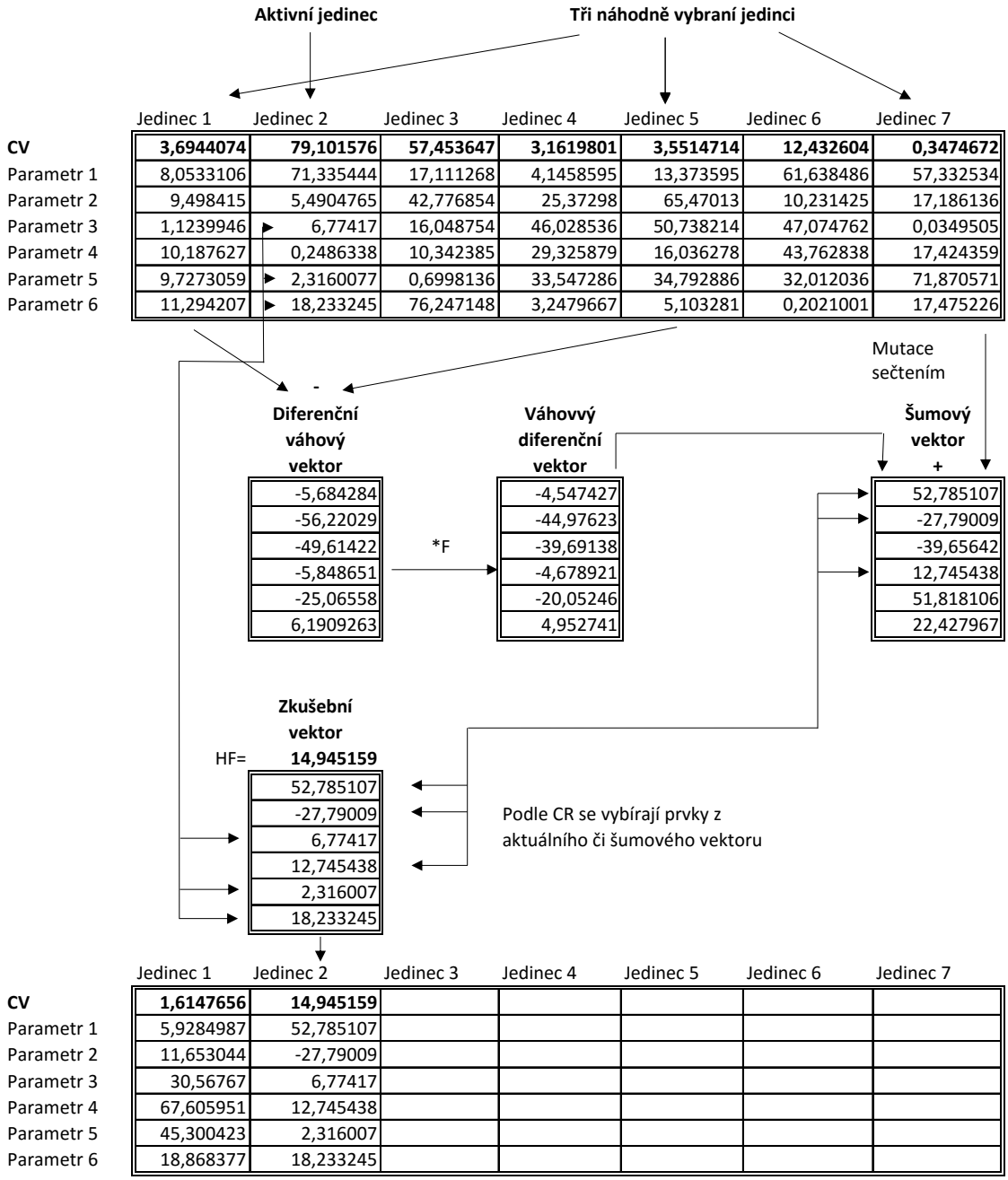
$$u_{i,j} = \begin{cases} v_{i,j} & pokud\ j = (n\ mod\ D), (n+1\ mod\ D), ..., (n+L-1\ mod\ D) \\ x_{i,j} & jinak \end{cases} \tag{8}$$

Při tvorbě každého zkušebního vektoru musí být čísla n a L nově zvolená dle způsobů uvedených výše.

4.4 Testovací funkce

Aby bylo možné zaznamenávat průběhy algoritmu DE, je nutné, aby byly dostupné testovací funkce, které budou sloužit jako účelová funkce. K tomu byly zvoleny funkce definované dle [20]. Z těchto 30 funkcí, tabulka 2, je možné generovat záznam z průběhu jednotlivých verzí DE.

Pro využívání testovacích funkcí je nezbytné mít také v adresáři společně s nástrojem vstupní data, využívaná k ohodnocování za pomoci těchto funkcí. Vstupní data úzce souvisí také s omezením kladeným na možné využívané dimenze, pro které lze data z běhů algoritmů získávat, jelikož jsou dostupné pouze pro dimenze rozsahu 2, 10, 20, 30, 50 a 100 [20]. Vstupní data jsou na médiu přiloženém k této práci.



Obrázek 3: Princip průběhu pro verzi DE/rand/1, řídicí parametry: D = 6, NP = 7, F = 0,8 a CR = 0,5 převzato z [39]

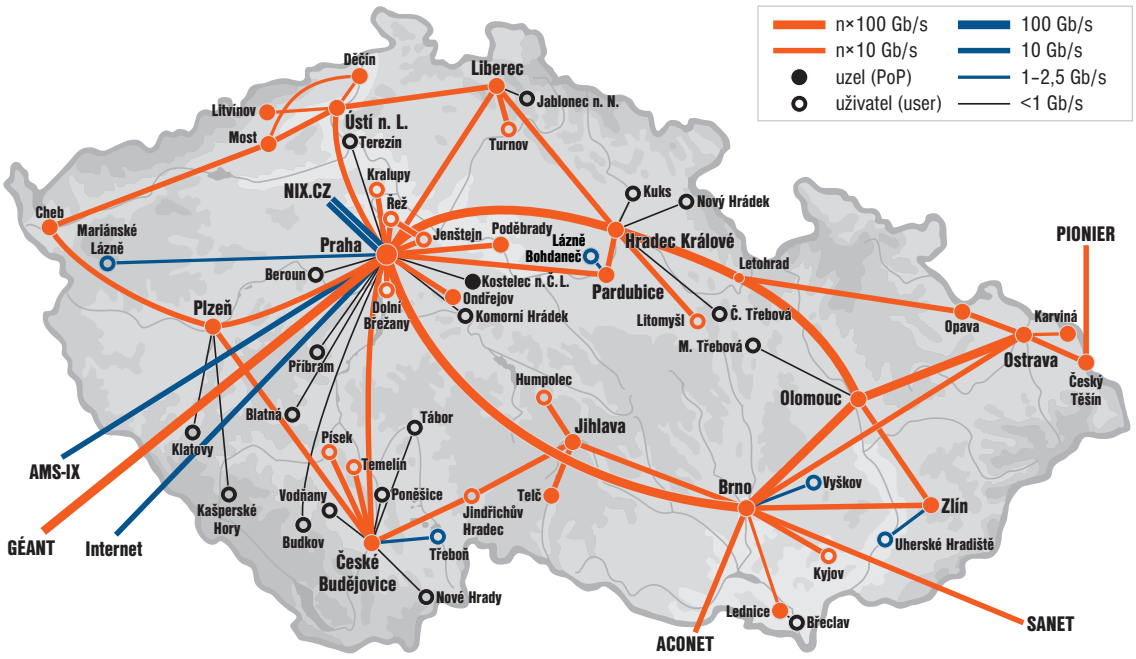
Tabulka 2: Použité funkce

Kategorie	Číslo	Funkce
Unimodální funkce	1	Rotated High Conditioned Elliptic Function
	2	Rotated Bent Cigar Function
	3	Rotated Discus Function
Jednoduché multi-modální funkce	4	Shifted and Rotated Rosenbrock's Function
	5	Shifted and Rotated Ackley's Function
	6	Shifted and Rotated Weierstrass Function
	7	Shifted and Rotated Griewank's Function
	8	Shifted Rastrigin's Function
	9	Shifted and Rotated Rastrigin's Function
	10	Shifted Schwefel's Function
	11	Shifted and Rotated Schwefel's Function
	12	Shifted and Rotated Katsuura Function
	13	Shifted and Rotated HappyCat Function
	14	Shifted and Rotated HGBat Function
	15	Shifted and Rotated Expanded Griewank's plus Rosenbrock's Function
	16	Shifted and Rotated Expanded Scaffer's F6 Function
Hybridní funkce	17	Hybrid Function 1 (N=3)
	18	Hybrid Function 2 (N=3)
	19	Hybrid Function 3 (N=4)
	20	Hybrid Function 4 (N=4)
	21	Hybrid Function 5 (N=5)
	22	Hybrid Function 6 (N=5)
Kompozitní funkce	23	Composition Function 1 (N=5)
	24	Composition Function 2 (N=3)
	25	Composition Function 3 (N=3)
	26	Composition Function 4 (N=5)
	27	Composition Function 5 (N=5)
	28	Composition Function 6 (N=5)
	29	Composition Function 7 (N=3)
	30	Composition Function 8 (N=3)

5 Sítě

Zjednodušeně lze říci, že sítí se rozumí seskupení bodů vzájemně propojených úsečkami. V matematické literatuře se sítě také někdy nazývají grafy. V terminologii sítí se úsečce spojující dva body říká hrana a jednotlivé body nazýváme uzly či vrcholy [22].

Sítě mohou být využity při reprezentaci mnohých fyzikálních, biologických či sociálních jevů [22], kde mohou pomoci s přehlednou interpretací dat. Za síť lze považovat například Internet, kde vrcholy představují jednotlivé počítače a za hrany lze považovat jejich datové propojení. Na obrázku 4 lze vidět znázornění topologie sítě CESNET2 platné k červenci 2016. Za vrcholy zde můžeme považovat jednotlivá města a hrany představují datové linky propojující tato města. Tato síť je také ohodnocená, což znamená, že jednotlivé hrany si nejsou rovny, ale jednotlivé hrany označují kvalitativně rozdílné datové linky. Kvalita datových linek je v obrázku znázorněna šíří hran.

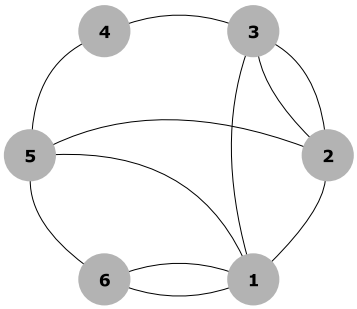


Obrázek 4: Topologie sítě CESNET2, červen 2016, převzato z [5]

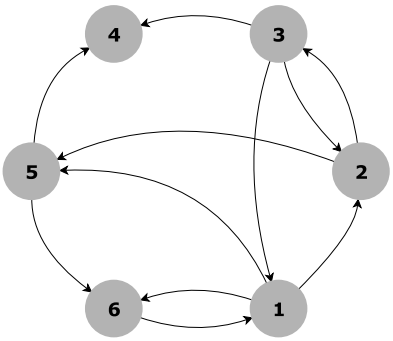
Nejjednodušším způsobem jak jednotlivé sítě využívat k vyhodnocování dat je přes jejich vizualizace. Vizualizace sítí se používá ve většině případů jako první krok při analýze sítě. Vizualizace dat může být velmi nápomocná, avšak velmi také záleží na velikosti dané sítě. Jestliže se vezme síť o velkém počtu vrcholů a hran, vizualizace jistě nebude pro lidské oko moc přehledná. Proto se vizualizace využívá spíše u menších sítí [22]. Na druhou stranu je potřeba analyzovat i tyto sítě, které nejdou přehledně vizualizovat, a proto potřebujeme jiné způsoby, jak data zkoumat. Z tohoto důvodu bylo v teorii sítí vyvinuto mnoho metrik, které pomohou porozumět rozsáhlým datovým strukturám.

5.1 Typy sítí

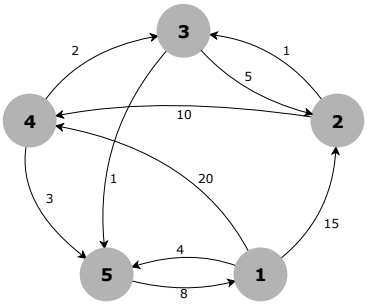
Jednoduché neorientované sítě poskytují informace pouze o tom zda hrana vrcholy propojuje či ne. Příklad neorientované sítě je znázorněn na obrázku 5. Pouze tyto informace by nám ale nemusely stačit na reprezentaci dat, kde například potřebujeme zohledňovat kvalitu určité hrany před jinou. V takovém případě můžeme každé hraně přiřadit hodnotu, která nám může pomoc při reprezentaci váhy, vzdálenosti či jakýchkoliv jiných veličin. Síť, kde se nacházejí ohodnocené hrany, nazýváme váženou, příklad na obrázku 7. U obrázku 4 je jako ohodnocení hrany použita propustnost datové linky. Dalším možným doplňkem hran může být jejich směr neboli orientace. Hranu s definovaným směrem nazýváme orientovanou hranou a síť, která obsahuje alespoň jednu orientovanou hranu se nazývá orientovaná, možno vidět na obrázku 6. Orientovanou hranu můžeme popsat jako uspořádanou dvojici vrcholů. V grafu se orientovaná hrana označuje šipkou ve směru její orientace.



Obrázek 5: Neorientovaná síť



Obrázek 6: Orientovaná síť



Obrázek 7: Vážená síť

5.2 Vlastnosti sítí

V této části budou popsány jednotlivé vlastnosti, které byly v dalších kapitolách využity pro vzájemné porovnání mezi sítěmi.

5.2.1 Délka průměrné nejkratší cesty

Pro neorientované, neválené grafy se jako délka cesty označuje počet hran spojující vrcholy i a j. Nejkratší cesta mezi těmito vrcholy je ta, která má minimální délku, jelikož cest může mezi dvěma vrcholy existovat více. Délka nejkratší cesty mezi vrcholy i a j se označuje d_{ij}. Tato definice lze použít také pro vážené grafy, avšak jako délka cesty se již nebere počet hran dané cesty, ale součet jejich vah. Pokud nastane situace, že se mezi vrcholy nenachází žádná cesta, pak $d_{ij} = \infty$.

Pro orientované grafy platí, že se ve většině případů liší nejkratší cesta z vrcholu i do vrcholu j od cesty z vrcholu j do vrcholu i, tedy $d_{ij} \neq d_{ji}$. Rovnice (9) definuje výpočet délky nejkratší

průměrné cesty sítě, kde N označuje počet jedinců v populaci [6].

$$l = \frac{1}{N(N-1)} \sum_{i \neq j} d_{ij} \tag{9}$$

5.2.2 Průměr grafu

Průměr grafu lze definovat jako délku nejdelší existující cesty mezi kterýmkoliv párem vrcholů v síti. Nevztahuje se tedy na cesty mezi vrcholy z dvou oddělených komponent, kdy je jejich vzdálenost považována za nekonečnou. U grafů skládajících se z více komponent lze průměr počítat pro každou komponentu zvlášť [22].

5.2.3 Koeficient shlukování pro jednotlivý vrchol

Koeficient shlukování pro jednotlivý vrchol (dále jen LCC) lze pro vrchol i definovat jako

$$LCC_i = \frac{(\textit{počet propojených párů sousedů vrcholu } i)}{(\textit{počet párů sousedů vrcholu } i)}, \tag{10}$$

kde je pro výpočet LCC použit počet jednotlivých párů sousedních vrcholů vrcholu i, které jsou vzájemně propojené, a tento počet je vydělen celkovým počtem párů v okolí vrcholu i, který lze získat jako $\frac{1}{2}k_i(k_i - 1)$, kde k_i představuje stupeň vrcholu i. Tato vlastnost je také někdy označována jako lokální koeficient shlukování a reprezentuje průměrnou pravděpodobnost, že pár sousedských vrcholů vrcholu i jsou také vzájemně sousedící. Neboli pravděpodobnost, že dva přátelé od jedince i jsou vzájemně také přátelé [22].

5.2.4 Globální koeficient shlukování

Výpočet LCC je také využíván v kontextu kalkulace globálního koeficientu shlukování (dále jen GCC), kde GCC je definován pro celou síť jako průměr LCC pro jednotlivé vrcholy, jak je vidět dle rovnice (11).

$$GCC = \frac{1}{N} \sum_{i \in N} LCC_i \tag{11}$$

5.2.5 Vážený koeficient shlukování

Ve vážených sítích lze jako rozšíření LCC považovat vážený koeficient shlukování (dále jen WCC). Tato vlastnost kombinuje topologickou informaci s informací o distribuci vah vrcholů v síti. WCC je definován rovnicí (12), kde se s_i rozumí síla vrcholu dle rovnice (17), k_i je stupeň vrcholu, w_{ij} váha hrany mezi vrcholy i a j a a_{ij} nabývá hodnoty 1 pokud existuje hrana mezi vrcholy i a j a hodnoty 0 pokud taková hrana neexistuje. Výraz $s_i(k_i - 1)$ lze interpretovat jako váhu každé hrany krát maximální počet tripletů, ve kterých se může vrchol i vyskytovat, a hraje zde roli

normalizačního faktoru, což zajišťuje, aby $0 \leq C_i^w \leq 1$.

$$LCC_i^w = \frac{1}{s_i(k_i - 1)} \sum_{j,h} \frac{(w_{ij} + w_{ih})}{2} a_{ij} a_{ih} a_{jh} \tag{12}$$

WCC představuje měřítko lokální soudržnosti, která bere v úvahu důležitost shlukovaných struktur na základě intenzity interakcí vyskytující se na lokálních tripletech. WCC počítá pro každý triplet, tvořený sousedskými vrcholy vrcholu i, váhu dvou hran mezi těmito vrcholy a vrcholem i. Tímto výpočtem dochází k zvážení, nejen počtu uzavřených tripletů v sousedství, ale také jejich relativní celkové váhy s přihlédnutím k síle vrcholu [2].

5.2.6 Stupeň vrcholu

Stupněm vrcholu v síti rozumíme počet hran k němu vedoucích. U neorientovaných sítí je stupněm vrcholu pouze jedno číslo. V orientovaných sítích má stupeň vrcholu dvě části a to výstupní stupeň pro hrany, které z daného vrcholu vycházejí, a vstupní stupeň pro hrany, které do daného vrcholu vstupují. Celkový stupeň vrcholu se poté u orientovaných sítí bere jako součet těchto dvou stupňů [4]. U příkladu, kde jako síť budeme uvažovat Internet, by se stupněm rozumělo počet datových linek vedoucích k jednotlivým počítačům či směrovačům. Výpočet stupně vrcholu lze provést dle rovnice (13) pro neorientovanou síť a dle rovnice (14) pro orientované sítě, kde k_i rozumíme stupeň vrcholu i a a_{ij} nabývá hodnoty 1, pokud existuje hrana mezi vrcholy i a j, nebo hodnoty 0, pokud taková hrana v síti neexistuje.

$$k_i = \sum_{j \in N} a_{ij} \tag{13}$$

$$k_i = k_i^{out} + k_i^{in} \tag{14}$$

Pro výpočet vstupního, respektive výstupního, stupně u orientovaných sítí lze postupovat dle rovnice (15), respektive dle rovnice (16).

$$k_i^{in} = \sum_{j \in N} a_{ji} \tag{15}$$

$$k_i^{out} = \sum_{j \in N} a_{ij} \tag{16}$$

5.2.7 Vážený stupeň vrcholu

Ve vážených sítích dochází k rozšíření stupně vrcholu na tzv. vážený stupeň, v některých publikacích označována také jako síla vrcholu [4]. Síla vrcholu je definována dle rovnice (17), kde se pod označením w_{ij} rozumí váha hrany z vrcholu i do vrcholu j. Síla vrcholu udává tedy součet vah všech hran s ním spojeným. Pro orientované sítě rozlišuje vstupní, respektive výstupní sílu

vrcholu, kde se bude jednat o součet vah všech vstupních hran do vrcholu, respektive všech výstupních hran.

$$s_i = \sum_{j \in N} w_{ij} \tag{17}$$

V tabulce 3 lze vidět vypočtené některé vlastnosti sítě zobrazené na obrázku 8.

5.2.8 Průměrný stupeň vrcholu

Výpočet stupně vrcholu, jak je definován rovnicí (13), se využívá při stanovení průměrného stupně vrcholu v síti. Jednoduše lze říct, že průměrný stupeň vrcholu v síti znamená průměr všech k_i, kde i nabývá hodnoty od 1 do N, N se rovná počtu vrcholů v síti. Jak je vidět podle rovnice (18) [6].

$$k = \frac{1}{N} \sum_{i \in N} k_i \tag{18}$$

5.2.9 Průměrný stupeň nejbližších sousedů vrcholu

Pokud se rozšíří okruh vrcholů, pro které je zkoumán jejich stupeň, na sousední vrcholy, lze získat metriku nazývanou jako průměrný stupeň nejbližších sousedů vrcholu [29]. Průměrný stupeň nejbližších sousedů vrcholu i lze definovat dle rovnice (19).

$$k_{nn,i} = \frac{\sum_{j \in N} a_{ij} k_j}{k_i} \tag{19}$$

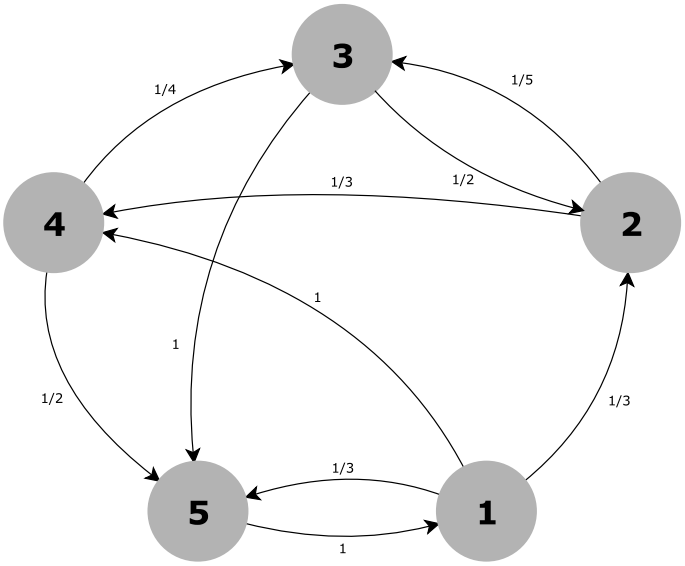
5.2.10 Průměrný vážený stupeň nejbližších sousedů vrcholu

Rozšířením rovnice (19) o počítání s váhou jednotlivých hran a sílou vrcholů, získáme rovnici pro výpočet váženého průměrného stupně nejbližších sousedů vrcholu, definovaného jako rovnice (20), kde s_i představuje sílu vrcholu, jinak řečeno vážný stupeň, dle rovnice (17) a w_{ij} značí váhu hrany mezi vrcholy i a j [29]. Tato vlastnost vyjadřuje míru vzájemných interakcí mezi sousedy s vysokým či nízkým stupněm [2].

$$k^w_{nn,i} = \frac{\sum_{j \in N} w_{ij} s_j}{s_i} \tag{20}$$

5.2.11 Distribuce stupňů

Se stupni vrcholů také úzce souvisí frekvence výskytu jednotlivých stupňů, neboli distribuce. Tato vlastnost je považována za jednu ze základních pro charakteristiku struktury sítě. Na hodnotu distribuce p_k jednotlivého stupně k v sítí můžeme nahlížet také jako na pravděpodobnost, s jakou bude mít náhodně vybraný vrchol právě tento stupeň k [22, 37]. Jedná se tedy o podíl počtu vrcholů se stupněm k a celkovým počtem vrcholů v síti, jak je vidět dle rovnice (21). U orientovaných sítí se rozlišuje distribuce pro vstupní a výstupní stupeň. Rozdíl při výpočtu



Obrázek 8: Orientovaná vážená síť

Tabulka 3: Stupně vrcholů a jejich síla pro síť na obrázku 8

Vrchol i	Stupeň k_i	Vstupní stupeň k_i^{in}	Výstupní stupeň k_i^{out}	Vstupní síla s_i^{in}	Výstupní síla s_i^{out}
1	4	1	3	1	1,67
2	4	2	2	0,83	0,53
3	4	2	2	0,45	1,5
4	4	2	2	1,33	0,75
5	4	3	1	1,83	1

je pouze v čitateli, kde se dosadí počet vrcholů se vstupním stupněm k^{in}, nebo počet vrcholů s výstupním stupněm k^{out}.

$$p_k = \frac{N_k}{N} \tag{21}$$

5.2.12 Betweenness centralita

V síti lze také posuzovat důležitost jednotlivého vrcholu podle počtu cest jím procházejících. Tato vlastnost se označuje jako betweenness centralita (dále jen BC). BC se nemusí týkat pouze vrcholů, ale může být vztažena ve stejném významu i na jednotlivé hrany sítě. Výpočet BC lze provádět dle rovnice (22), kde $\sigma(i, u, j)$ představuje počet nejkratších cest mezi vrcholy i a j, které zároveň procházejí skrz vrchol nebo hranu u. Výraz $\sigma(i, j)$ je počet všech nejkratších cest

mezi vrcholy i a j. Výsledkem je součet podílů pro každou jednotlivou dvojici vrcholů sítě [6].

$$C_{BC,u} = \sum_{ij} \frac{\sigma(i,u,j)}{\sigma(i,j)} \tag{22}$$

5.2.13 Closeness centralita

Další z vlastností se zaměřuje na vzdálenost z vrcholu do ostatních vrcholů. Jedná se o closeness centralitu (dále jen CC). Dle rovnice (23) je zřejmé, že se jedná o podíl počtu vrcholů v sítí snížených o 1 a součtu délky cest z vrcholu i do všech ostatních vrcholů [29].

$$C_{CC,i}^{-1} = \frac{n-1}{\sum_{j \in N, j \neq i} d_{ij}} \tag{23}$$

5.2.14 Eigenvector centralita

Za rozšíření jednoduché vlastnosti jako je stupeň vrcholu lze považovat eigenvector centralitu (dále jen EC). Na stupeň vrcholu lze nahlížet jako na ohodnocení vrcholu, kdy za každý sousední vrchol obdrží jeden bod. Avšak ne všechny sousední vrcholy jsou si rovny. Za mnoha okolností důležitost vrcholu v síti stoupá s počtem propojení na ostatní vrcholy a jejich propojení dál v síti. Na tomto konceptu se zakládá právě EC, místo pouhého ohodnocení dle počtu sousedů je u EC každý vrchol ohodnocen proporcionálně vzhledem k součtu skóre EC jeho sousedů [22]. Vyjádřeno dle rovnice (24), kde a_{ij} nabývá hodnoty 1 pokud existuje hrana mezi vrcholy i a j a hodnoty 0 pokud taková hrana neexistuje.

$$C_{EC,i} = \sum_{j} a_{ij} C_{EC,j} \tag{24}$$

6 Nástroj pro generování průběhu DE

Tento nástroj byl implementován v jazyce C++ [36] jako konzolová aplikace a slouží k zaznamenávání průběhů různých verzí algoritmu DE. Při vývoji bylo používáno vývojové prostředí Microsoft Visual Studio 2015.

Jazyk C++ byl k tomuto účelu zvolen z důvodu již dostupných testovacích funkcí, které jsou právě v tomto jazyce implementovány. Konkrétně se jedná o testovací funkce dle [20]. Jazyk C++ je následovník jazyka C, kde využívá jeho vlastnosti jako přenositelnost a výkonnost, které dále rozšiřuje o objektově orientované vlastnosti [27].

6.1 Popis nástroje

Jak již bylo zmíněno výše nástroj je implementován v jazyce C++, proto není vázán na konkrétní operační systém. Je jej tedy po kompilaci zdrojový kódů možné používat jak na systémech Microsoft Windows, tak MacOS či Linux. Zdrojové kódy nástroje jsou k dispozici na přiloženém médiu. Nástroj je uzpůsobený pro spouštění z příkazové řádky s argumenty. Mezi tyto argumenty patří dimenze problému, počet generací pro běh DE, verze algoritmu DE, výběr účelové funkce a počet běhů. Tyto argumenty lze volitelně zadat při spuštění programu, pokud nejsou zadány, nebo jsou zadány nekompletně dojde ke spuštění nástroje s výchozími parametry dle tabulky 5.

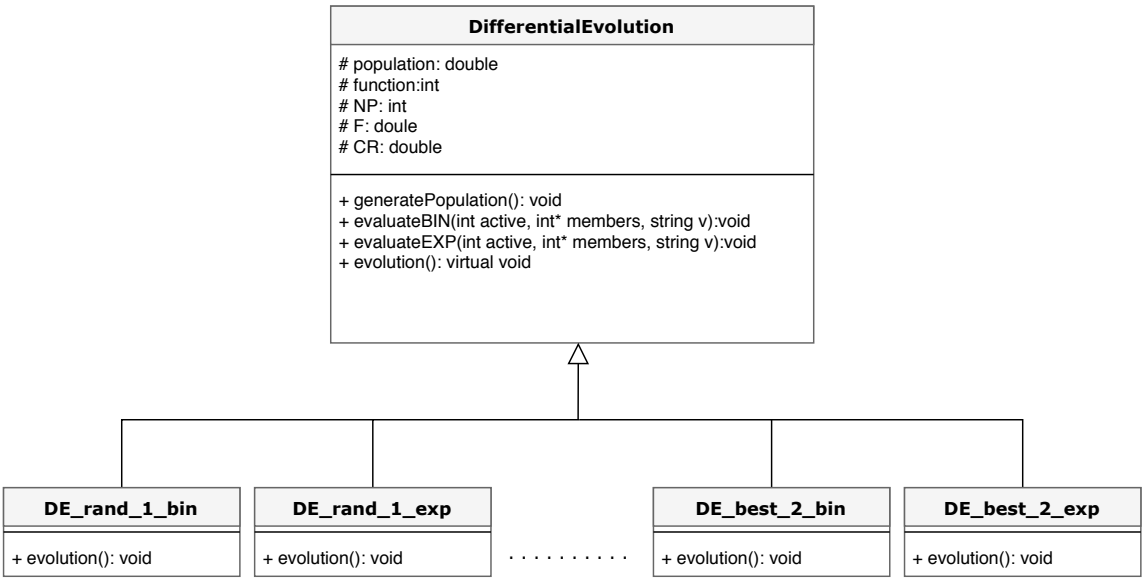
Tabulka 4: Implementované verze algoritmu DE

Číslo	Verze
1	DE/rand/1/bin
2	DE/rand/2/bin
3	DE/best/2/bin
4	DE/best/1/bin
5	DE/current-to-best/1/bin
6	DE/rand/1/exp
7	DE/best/1/exp
8	DE/rand/2/exp
9	DE/best/2/exp
10	DE/current-to-best/1/exp

Tabulka 5: Hodnoty výchozích parametrů nástroje

Parametr	Výchozí hodnota
Dimenze problému	10
Počet generací	3000
Verze algoritmu	1 - 10
Účelové funkce	1 - 30
Počet běhů	500

Seznam účelových funkcí, včetně jejich čísla využívaného pro parametr spuštění lze najít v tabulce 2. Na výpise 2 lze vidět příklad spuštění programu společně se zadáním argumentů, kdy byl algoritmus spuštěn na dimenzi 2, s počtem generací 5000, účelová funkce 1, tedy rotated high conditioned elliptic function, verze algoritmu DE/rand/1/bin, dle tabulky 4, a počet běhů stanoven na 250. Po spuštění program vypíše zadané parametry, aby je uživatel mohl zkontrolovat a následuje generování samotných logů. Program postupně informuje o průběhu



Obrázek 9: UML diagram části nástroje pro zaznamenávání běhu algoritmu DE

zobrazováním aktuálně probíhající verze algoritmu DE, zvolené účelové funkci, nejlepší dosažené hodnoty účelové funkce a doby jak dlouho aktuální běh trval v sekundách.

```
C:\>DE_VS2013.exe 2 5000 1 1 250
```

Výpis 2: Spuštění nástroje z příkazové řádky

6.2 Implementační detaily

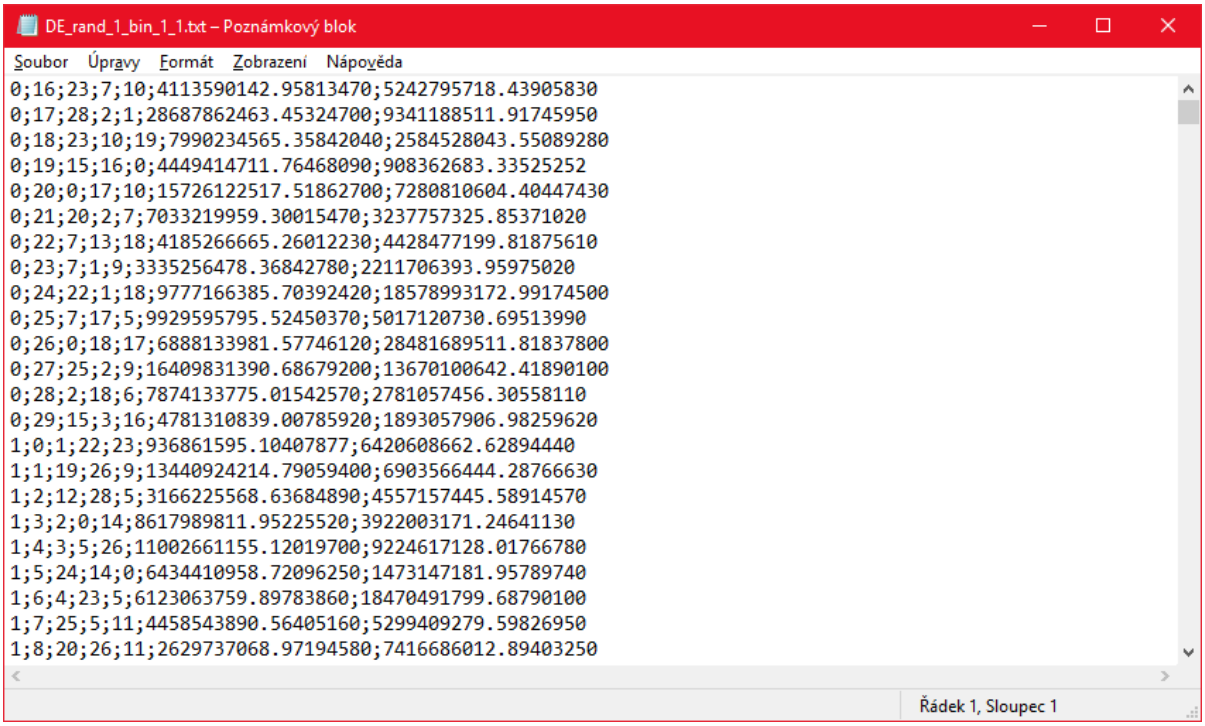
Po implementační stránce je možné rozdělit program do 3 hlavních částí. První část zabývající se samostatnou obsluhou programu, spouštěním implementací různých verzí algoritmu DE a základním zpracováním a řízením toku programu, ať už dle parametrů definovaných uživatelem, nebo dle nastavených výchozích hodnot. Za druhou část lze považovat implementace jednotlivých verzí algoritmu DE. Těchto verzí je v tomto nástroji implementováno celkem 10, konkrétně verze v tabulce 4, a to pro 5 nejběžněji využívaných mutačních strategií [7], každou z nich pro binomické i exponenciální křížení. Ve struktuře programu je každá tato verze implementována jako samostatná třída. Všechny tyto třídy mají jednoho společného předka, který zajišťuje například nastavení řídicích parametrů, které jsou pro všechny verze totožné. Tato struktura je znázorněna UML diagramem na obrázku 9. Poslední, tedy třetí částí, jsou naimportované knihovny obsahující implementace účelových funkcí.

6.3 Podoba a význam exportovaných dat

Hlavním úkolem toho nástroje je export detailního průběhu jednotlivých běhů různých verzí algoritmu DE. Exportovaná data jsou dále využívány k analýze a porovnání těchto běhů. Například k zjištění různých závislostí mezi běhy, které dosáhly co nejlepších výsledků, aby bylo možné dále běhy algoritmů optimalizovat, třeba různým nastavením parametrů. Soubory, které jsou tímto nástrojem generovány, jdou rozdělit do dvou skupin.

První soubor je vždy jeden pro každou verzi algoritmu DE, má název ve tvaru DE_x_y_z.txt, a obsahuje informace o všech uskutečněných bězích právě této verze. Obsahově se jedná o typ csv souboru [32], což znamená, že jednotlivé informace jsou odděleny jednotným oddělovačem, v tomto případě je použit znak středník a pro oddělení jednotlivých záznamů je využito ukončení řádku. Každý řádek obsahující jeden záznam se skládá z těchto informací: verze algoritmu, čísla účelové funkce, počtu jedinců pro generaci, celkový počet generací, nejlepší nalezený výsledek účelové funkce a název souboru, ve kterém je zaznamenán detailní průběh daného běhu. Tímto je tedy nastíněn i obsah druhého typu souboru.

Druhý typ souboru je specifický pro každý jednotlivý běh, název má podobný jako předchozí soubor a to ve tvaru DE_x_y_z_a_b.txt, kde část DE_x_y_z označuje verzi algoritmu, hodnota označená jako 'a' značí číslo účelové funkce, na které byl daný běh proveden, a hodnota 'b' označuje pořadí souboru v rámci jedné verze a jedné účelové funkce tak, aby nedocházelo k vzájemnému přemazávání souborů. Jednotlivé hodnoty záznamu jsou opět odděleny středníkem a každý řádek nese informace týkající se jednoho záznamu. Tento soubor je generován, aby bylo možné detailně analyzovat průběh běhu celého algoritmu. Každý řádek se týká právě jednoho aktivního jedince, pro kterého je zaznamenán jeho přínos v dané generaci. Postupně jeden záznam obsahuje tyto informace: aktuální generace, číslo aktivního jedince, poté následují čísla jedinců se kterými aktivní jedinec interagoval, jejich počet závisí na verzi algoritmu, a poslední dvě hodnoty značí hodnoty ohodnocení jedinců účelovou funkcí, první v pořadí je hodnota účelové funkce pro aktivního jedince, druhá v pořadí je hodnota účelové funkce nově vytvořeného jedince. Příklad formátu tohoto souboru je vidět na obrázku 10. Tento soubor je také využíván pro převod běhu na sít a analýzu v nástroji popsanému v následující kapitole.



Obrázek 10: Ukázka expotovaných dat o průběhu algoritmu DE

7 Nástroj pro převod běhů na síť a následnou analýzu

Na rozdíl od prvního nástroje, psaného v jazyce C++, byl pro tento nástroj zvolen programovací jazyk JAVA. Jedním z hlavních důvodů proč padl výběr právě na tento programovací jazyk je dostupnost knihovny JUNG [14], která obsahuje třídy pro reprezentaci sítí a také knihovny pro výpočet některých vlastností, což velmi usnadnilo vývoj. Vývoj probíhal ve vývojovém prostředí NetBeans, ve spolupráci s nástrojem pro tvorbu grafického rozhraní JavaFX Scene Builder. Nástroj slouží pro převod a analýzu běhů algoritmu DE na síť, vizualizaci sítě, výpočtu vlastností a možnosti vzájemného porovnání dvou skupin těchto běhů. Například lze tímto nástrojem porovnávat odlišnosti ve vlastnostech, mezi běhy s nejlepším výsledkem a těmi s výsledkem nejhorším, a následně vyhledat závislosti, díky kterým můžeme zefektivnit nastavení jednotlivých verzí algoritmu DE. Jako vstupní data tento nástroj využívá generované logy z předchozího programu.

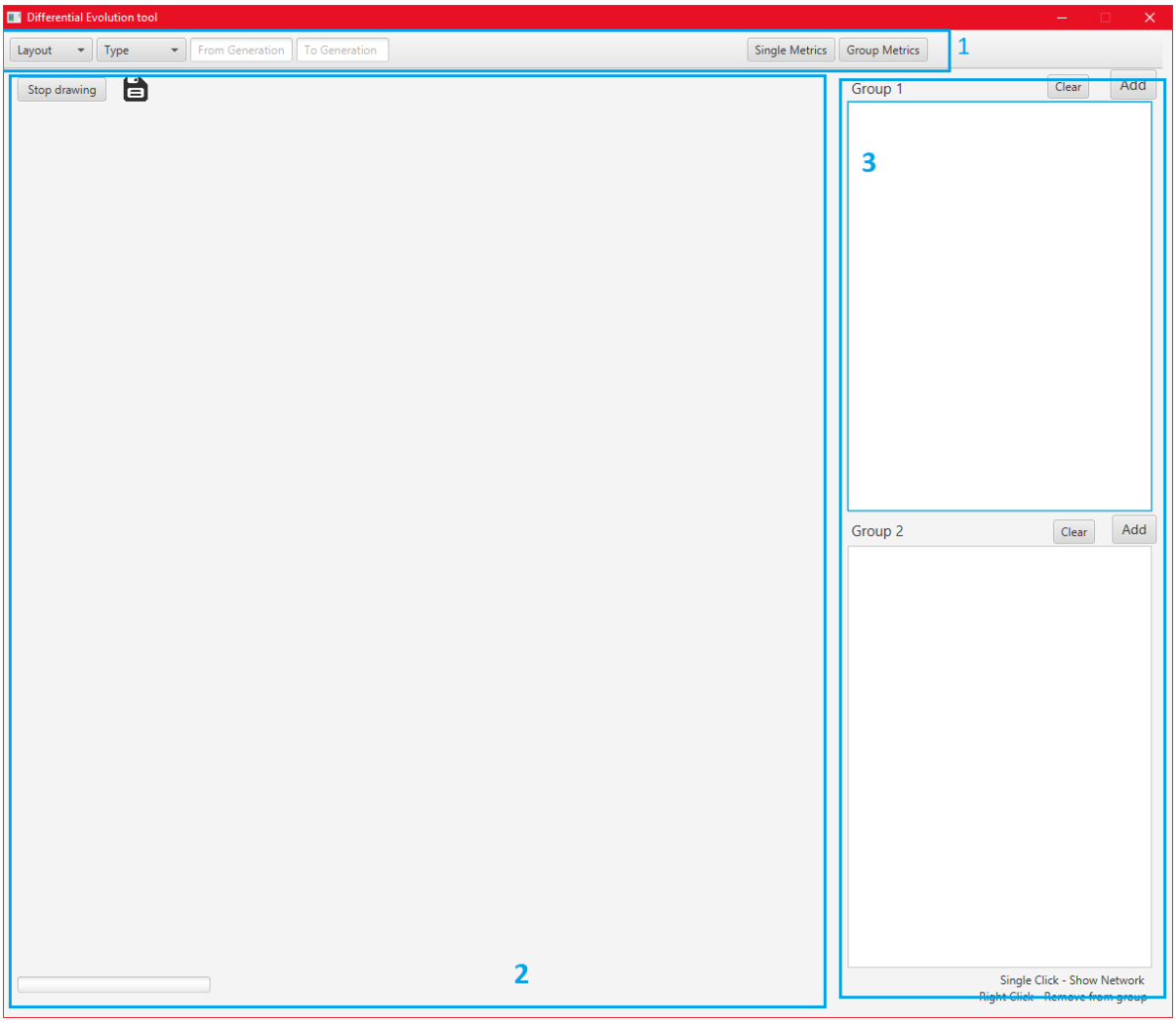
7.1 Instalace a spuštění

Samotný nástroj je dostupný na médiu přiloženém k této práci, včetně zdrojových kódů. Avšak aby bylo možné jej využívat je třeba mít na počítači nainstalovaný Java Development Kit, který lze stáhnou ze stránek [24]. Po stažení je potřeba ještě instalace a pokud vše proběhne bez problému je možné začít nástroj využívat.

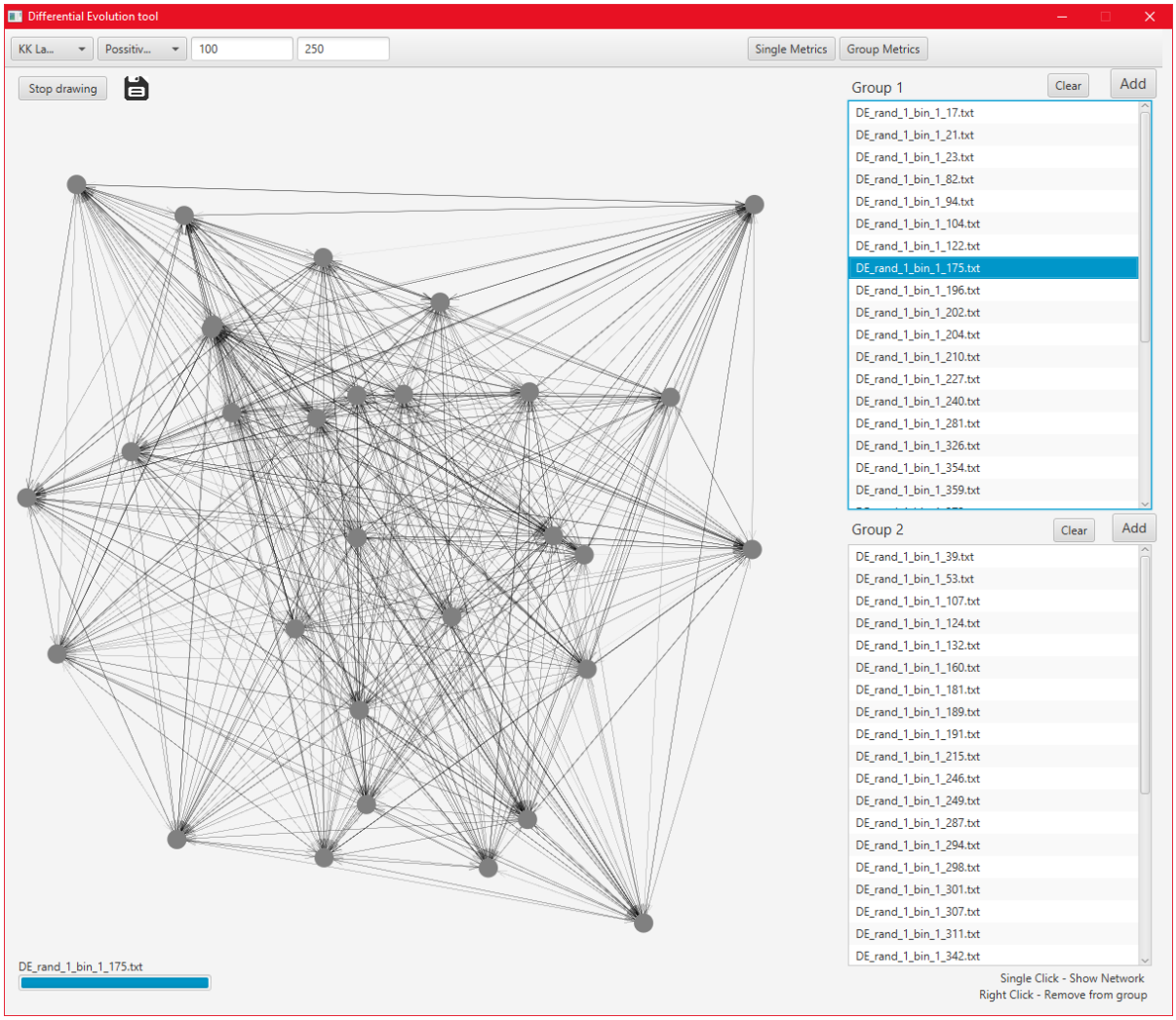
7.2 Popis grafického rozhraní a ovládání nástroje

Po spuštění programu se uživateli zobrazí úvodní obrazovka jako na obrázku 11. Tuto obrazovku jde rozdělit do tří hlavních částí, které si nyní popíšeme. První část, na obrázku 11 označená modrou číslicí 1, představuje menu programu. Toto menu lze rozdělit dále do levé a pravé části, levá část se týká ovládání vizualizace sítí, kde si může uživatel nastavit z rozbalovací nabídky rozložení vykreslených vrcholů nebo typ převodu, který je použit pro vykreslenou síť. Další dva boxy v této části slouží pro specifikaci hodnot od-do, které určují rozsah generací, pro které dochází k vizualizaci. Nyní se přesuneme do pravé části menu. Zde uživatel nalezne dvě tlačítka, první slouží pro otevření okna programu, kde je možno analyzovat vlastnosti jedné zvolené sítě. Druhé tlačítko slouží pro vzájemné porovnávání vlastností jednotlivých sítí obsažených ve dvou skupinách.

Druhou částí, tou která zabírá největší plochu úvodní obrazovky, je plátno, na kterém dochází k samotné vizualizaci převedených sítí. V této části si lze vlevo dole povšimnout ukazatele průběhu vykreslování, který postupně informuje uživatele o stavu vykreslování, jelikož u rozsáhlých sítí může tato vizualizace trvat delší dobu. V levém horním rohu této části proto také vidíme tlačítko k zastavení vykreslování a hned vedle něj také ikonu pro uložení. Po kliknutí na tuto ikonu je možné právě vykreslenou síť uložit do png souboru, cestu a název souboru si



Obrázek 11: Úvodní obrazovka nástroje pro převod a analýzu



Obrázek 12: Úvodní obrazovka nástroje pro převod a analýzu - včetně vizualizace

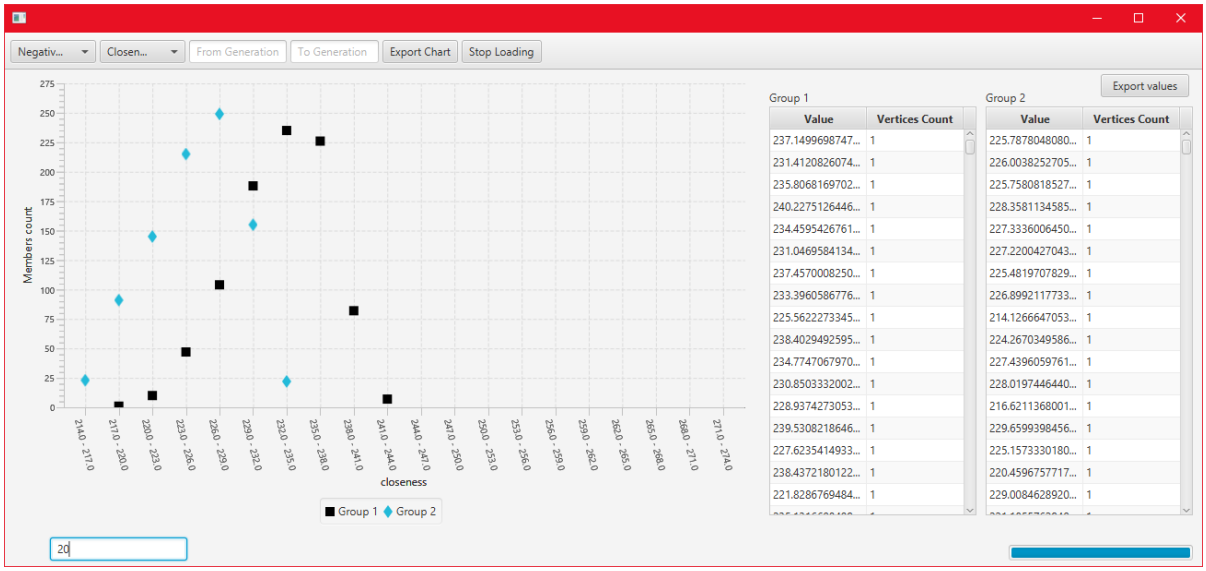
může uživatel zvolit. Při vykreslování vážených sítí je zohledněna váha jednotlivých hran, která je znázorněna různou šířkou hran.

Třetí částí úvodní obrazovky, je prostor, kde uživatel vidí načtené soubory pro převod na síť. Tato část je rozdělena do dvou skupin, pro každou skupinu je dostupné tlačítko pro přidání souboru, po kliknutí je uživatel vyzván k zadání cesty k souboru, který chce přidat, dále pak tlačítko pro smazání všech dosud přidělených souborů do dané skupiny. Pokud uživatel klikne levým tlačítkem myši na název souboru, dojde k jeho převodu na síť a následné vizualizaci na již výše zmíněném plátně. Pro vykreslení jsou zvoleny výchozí hodnoty, které uživatel poté může měnit pomocí menu v části číslo 1. Pokud uživatel chce již přidaný soubor do skupiny odstranit, může tak učinit kliknutím pravým tlačítkem myši na jeho zobrazený název. Rozdělení do skupin, které zde uživatel učiní, je dále využito pro vzájemné porovnávání vlastností. Ukázka vzhledu programu po nahrání souborů a převod jednoho z nich na síť včetně vizualizace je vidět na obrázku 12.

Obrázek 13: Obrazovka pro výpočet vlastností jedné sítě

Pokud má uživatel zájem o výpočet a zobrazení vlastností pouze pro jednu síť, musí soubor, pro který chce provést převod na síť a následnou analýzu, nejprve přidat do jedné ze dvou skupin. Poté stačí aby soubor s vybranými daty vybral, dojde k jeho označení, následnému převodu na síť a vykreslení. Dále může uživatel kliknout v menu na tlačítko pro vlastnosti jedné sítě a dojde k otevření nového okna, kde uživatel může dále zvolit typ převodu na síť a konkrétní vlastnost kterou chce vypočítat. Dále lze také specifikovat rozsah generací. Toto okno je zobrazeno na obrázku 13, kde byla již byl zvolen jako typ převodu pozitivní interakce, vlastnost vstupní stupeň vrcholu a rozsah generací 0 - 5. Hlavní dominantou této obrazovky je histogram, na kterém jsou vypočtené vlastnosti vizualizovány pro lepší přehlednost. Vpravo od histogramu je vidět tabulka, ve které jsou pro všechny vrcholy zobrazeny jejich hodnoty pro danou vlastnost. Dále má možnost uživatel exportovat tyto data nebo uložit histogram. V levém dolním rohu obrazovky se nachází box, díky kterému může uživatel specifikovat počet kategorií histogramu na ose x.

Hlavní myšlenkou tohoto nástroje bylo vzájemné porovnání dvou skupin a zjištění rozdílu u jejich vlastností. K tomuto slouží okno, které je z úvodní obrazovky dostupné pod tlačítkem skupinové metriky. Před tím však uživatel musí načíst alespoň jeden soubor s daty do jedné ze dvou skupin. Pokud tak učiní, zobrazí se mu nové okno jako na obrázku 14. Tato obrazovka je velmi podobná té pro analýzu jedné sítě, avšak počet zde zobrazených tabulek se liší. Nachází se zde dvě tabulky, kdy každá představuje hodnoty, které nabývají vrcholy v jedné či druhé skupině. Už zde není ke každému vrcholu přiřazena hodnota, kterou pro danou vlastnost nabývá, ale jsou zde zobrazovány hodnoty a k nim přiřazen počet vrcholů nabývající tuto hodnotu. Opět je možné



Obrázek 14: Obrazovka pro porovnání vlastností mezi 2 skupinami

uložit histogram a exportovat vypočtené hodnoty. Pokud se uživatel rozhodne uložit vypočtené hodnoty, výsledný soubor nemá stejnou podobu jako je vidět na obrázku 14. V souboru jsou uloženy hodnoty pro jednotlivé vrcholy a to odděleně pro první a druhou skupinu.

7.3 Implementační detaily

Program lze z pohledu implementace rozložit na tři části. První část se stará o uživatelskou přívětivost a představuje grafické rozhraní aplikace. K tomuto účelu bylo využito platformy JavaFX. Tato platforma poskytuje plugin pro vývojové prostředí NetBeans a využívá tzv. FXML souborů. FXML soubory staví na bázi XML a jsou využívány pro tvorbu uživatelského rozhraní v JavaFX aplikacích. S těmito soubory lze interaktivně pracovat pomocí nástroje JavaFX Scene Builder [26]. Tímto způsobem byla navrhnuta grafická stránka programu. O řízení toku programu se starají kontroléry jednotlivých oken a využívají k tomu implementované třídy, viz další odstavec.

Jako další část programu lze pojmout implementaci jednotlivých tříd. Podle účelu lze tyto třídy rozdělit do skupin. První skupina se stará o načítání souborů vygenerovaných prvním nástrojem, obsahující záznamy z průběhu algoritmu, a jejich převodem na síť. Těchto převodů je implementováno celkem osm. Jedná se o

- pozitivní interakci,
- pozitivní váženou interakci,
- negativní interakci,
- negativní váženou interakci,

- postupný vývoj vrcholů,
- pozitivní vážená interakce s prvky mravenčí optimalizace,
- negativní vážená interakce s prvky mravenčí optimalizace,
- nejlepší z generace,

kde jednotlivým převodům bude věnována samostatná kapitola. Za další skupinu můžeme považovat třídy starající se o výpočet jednotlivých vlastností. Z této skupiny část tříd využívá k výpočtům již implementované části knihovny JUNG a část tříd byla pro výpočet vlastností implementována samostatně. Všechny tyto třídy starající se o výpočet vlastností mají společného předka v podobě abstraktní třídy. Poslední větší skupinou tříd jsou třídy starající se o grafické znázornění daných vlastností. Tyto třídy jsou využívány v grafickém rozhraní pro tvorbu histogramů. Schématické znázornění vztahů mezi třídami je zobrazeno UML diagramem na obrázku 15.

Za poslední část, ze tří kategorií rozčlenění programu, lze považovat importovanou knihovnu JUNG [14]. Z této knihovny byly hlavně využity třídy pro reprezentaci jednotlivých sítí, avšak bylo k nim nutné implementovat vlastní třídy představující vrchol a hranu, kde bylo nutné implementovat rozhraní, které poskytuje knihovna JUNG. Dále jak již bylo naznačeno obsahuje tato knihovna implementované třídy pro výpočet některých vlastností, konkrétně pro EC, CC, BC, LCC a stupeň vrcholu, včetně vstupního a výstupního.

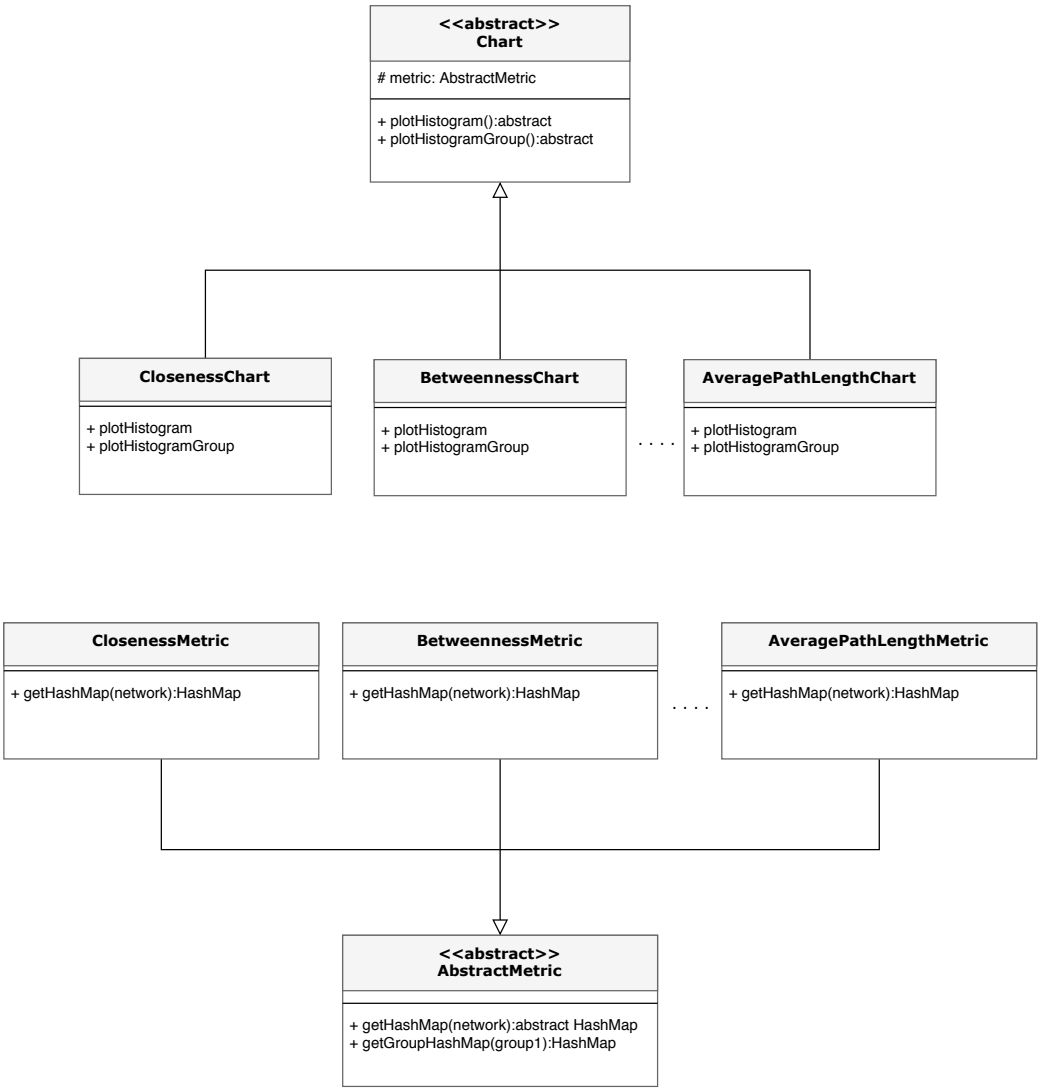
7.4 Typy převodů na síť

Důležitou funkcí tohoto nástroje je převod získaných dat, z jednotlivých běhů algoritmu DE, na síť. Důvodem tohoto převodu je možná vizualizace těchto dat a také výpočet charakteristických vlastností pro síť, které mohou sloužit k analýze a vzájemnému porovnávání. Nástroj disponuje celkem 8 různými převody. Ve všech případech převodu je vzniklá síť orientovaná. V těchto převodech jsou jedinci z populace představováni v síti jako vrcholy a interakce mezi nimi pomocí hran.

7.4.1 Převod pozitivní interakce

Prvním převod je nazvaný pozitivní interakce. V tomto převodu je snaha o zachycení situací, kdy při běhu algoritmu došlo k vylepšení stávajícího jedince, tedy došlo k vzniku nového jedince s lepším ohodnocením účelovou funkcí. Pokud tato situace nastane, jsou přidány orientované hrany mezi vrchol představující aktivního jedince a vrcholy představující jedince se kterými interagoval.

Při analýze sítě vzniklé tímto převodem značí tedy výstupní stupeň vrcholu počet jedinců se kterými interagoval a došlo k vylepšení, naopak vstupní stupeň vrcholu značí počet interakcí, kterých se daný jedinec účastnil. Čím vyšší průměrný stupeň vrcholu v této síti, tím více interakcí



Obrázek 15: UML diagram části nástroje pro převod a analýzu

v tomto běhu probíhalo. Také pokud se zaměříme na vlastnost průměrný stupeň sousedů, tak je možné zjistit, zdali jedinci s vyšším stupněm interagovali častěji mezi sebou. Vlastnosti počítající s váhou hran, nemají u tohoto převodu příliš vypovídající hodnotu, jelikož u většiny z nich dochází k překrytí s jinými, například stupeň vrcholu nabývá stejných výsledků jako síla vrcholu, jelikož u nevážených sítí je váha každé hrany 1. Další vlastností, na kterou se můžeme zaměřit, je distribuce jednotlivých stupňů vrcholů, zde můžeme odhalit, zdali docházelo k interakci mezi vrcholy rovnoměrně, či se v síti nachází mnoho vrcholů, u kterých nedošlo ke vzniku lepších jedinců.

Další možné vlastnosti se zaměřují na pozici vrcholu v rámci celé sítě, první z nich je BC. Vrchol s vyšší hodnotou BC je v síti důležitější z pohledu informací skrz něj putujících [22], z pohledu algoritmu DE, můžeme říct, že skrz tohoto jedince byl genom předán dále vícekrát, než u jedince s nižší hodnotou BC. CC představuje vzdálenost z vrcholu do ostatních vrcholů [29], u tohoto typu převodu tedy značí, jestli vrchol interagoval s ostatními vrcholy přímo. EC zahrnuje do porovnání vlivu vrcholu v síti i jeho sousedy [23], v aktuálním kontextu lze tedy dle EC vysledovat jak úspěšně mohl daný jedinec předávat svůj genom. Nahlížet na to lze tak, že pokud jedinec předal svůj genom jednomu ze sousedních vrcholů a ten poté velké většině ostatních, zahrne se to i do hodnoty EC původního jedince.

Vlastnosti LCC a GCC v tomto případě reflektují tzv. tranzitivitu sítě. Pokud jedinec x_1 interagoval s jedincem x_2 a jedinec x_2 interagoval s jedincem x_3, tak jestli došlo ke vzájemné interakci i mezi jedinci x_1 a x_3 [23]. LCC tedy ukazuje, zdali docházelo ve větší míře k výskytu skupin jedinců, kteří interagovali vzájemně. Délka průměrné nejkratší cesty u tohoto převodu značí přes kolik průměrně jedinců putoval genom z jednoho do druhého, nově vytvořeného. Průměr grafu zde představuje nejdelší možnou cestu genomu mezi dvěma jedinci.

7.4.2 Převod pozitivní vážená interakce

Druhý způsob převodu pozitivní vážená interakce. Jedná se v podstatě o rozšíření předešlého o vážené hrany. Váha hran je zde určena dle počtu interakcí mezi jedinci, kdy za každou interakci je váha hrany povýšena o 1. Z tohoto způsobu vychází, že váha hrany představuje počet interakcí mezi danými jedinci, které tato hrana propojuje. Aby však docházelo k reflexi skutečnosti v tom ohledu, že z pohledu sítí je nejlepší hrana ta s nejmenší váhou (cenou), je pro výpočet vlastností využita převrácená hodnota váhy hrany, tedy $\frac{1}{w_{ij}}$, pro hranu mezi vrcholy i a j.

Vlastnosti sítě vzniklé tímto převodem se od předcházejícího způsobu liší pouze v případech, že daná vlastnost zohledňuje váhy hran. Jednou z těchto metrik je síla vrcholu. Zde bude výstupní síla vrcholu reflektovat počet nově vzniklých jedinců z aktivního jedince, kde menší hodnota značí více, a vstupní síla vrcholu reflektuje počet nově vzniklých jedinců, kteří byli vytvoření v interakci s daným vrcholem, opět menší hodnota značí více vzniklých jedinců. Vážený průměr stupně nejbližších sousedů vyjadřuje jak moc došlo k interakcím mezi sousedy s vysokým či nízkým stupněm.

7.4.3 Převody negativní interakce a negativní vážené interakce

Třetí a čtvrtý způsob převodu navazuje na předchozí dva, jedná se o negativní interakci a negativní váženou interakci. Tyto převody sledují vznik nových jedinců, kdy nově vzniklý jedinec dosahoval horšího ohodnocení účelovou funkcí a nedošlo tedy k nahrazení původního jedince novým. Pokud dojde k takové situaci jsou v síti přidány hrany z vrcholu, který představuje aktivního jedince, do vrcholů se kterými aktuálně interagoval a nedošlo ke vzniku jedince lepšího. Na tomto typu převodů lze tedy sledovat kolik proběhlo interakcí, kdy nedošlo k posunu při běhu algoritmu k lepšímu výsledku.

7.4.4 Převody pozitivní a negativní vážené interakce s prvky mravenčí optimalizace

Další dva z převodů vycházejí z již výše zmíněných, a to konkrétně z pozitivní a negativní vážené interakce. Jedná se o pozitivní a negativní váženou interakci s prvky mravenčí optimalizace. Hlavním rozdílem je zde využití prvku známého z mravenčích algoritmů [9]. Tento prvek se týká způsobu ohodnocování hran, kdy na konci každé generace dochází k ponížení váhy každé hrany o určitou konstantu, v tomto případě o 1 %.

Tento způsob vychází z chování mravenců, kteří při pátrání po potravě své úspěšné cesty označují feromony, tak aby je bylo možné využít dalšími. Postupně tedy dochází k zvyšování atraktivity cesty dle počtu mravenců ji využívající. U feromonů časem dochází také k postupnému vypařování, čímž dochází ke snižování atraktivnosti méně využívaných cest pro ostatní mravence [19]. Díky této vlastnosti dochází k postupnému snižování významu interakcí použitých v minulosti a zvyšuje se význam těch aktuálních a častěji využívaných.

7.4.5 Převod postupný vývoj vrcholů

U dalšího způsobu převodu je třeba upozornit, že při použití dochází k vytváření velmi rozsáhlých sítí, které představují značnou paměťovou náročnost a paměť počítače nemusí být dostatečná pro některé výpočty prováděné knihovnou JUNG. Jedná se o převod nazvaný postupný vývoj vrcholů, kdy principem je, že dochází k přidání nového vrcholu představujícího nově vzniklého jedince. Hrany zde představují předky, ze kterých nový jedinec vznikl a jsou orientovány směrem do něj. Pro ukázku rozsáhlosti sítě vzniklé tímto způsobem, pro běh algoritmu s počtem jedinců 30 a počtem generací 3000, může počet vrcholů sítě být až 90 000.

7.4.6 Převod nejlepší z generace

Poslední způsob převodu spočívá ve sledování vývoje pouze nejlepšího jedince z generace a toho, kteří jedinci se na vzniku tohoto jedince podíleli. Při převodu je tedy v první fázi vytvořen vrchol představující každého jedince a zjištění hodnoty účelové funkce u nejlepšího z nich. Poté pokud v aktuální generaci dojde k vytvoření jedince s lepším ohodnocením účelovou funkcí, je

vytvořen nový vrchol, který jej představuje a jsou doplněny hrany vedoucí z vrcholů, ze kterých tento nově vytvořený potomek získal části svého genomu.

8 Analýza a vyhodnocení dat

Tato sekce se bude zabývat analýzou a vyhodnocením dat získaných z nástrojů popsaných výše. Z důvodu velké časové náročnosti generování logů z běhu algoritmu DE a následnému výpočtu vlastností, bude ukázka analýzy dat provedena pro jednu verzi, touto zvolenou verzí algoritmu DE je DE/rand/1/bin. Řídící parametry byly nastaveny dle tabulky 6. Z celkového počtu 500 běhů bylo následně vybráno 60, dvě skupiny po 30, pro 30 nejlepších a 30 nejhorších běhů, u kterých je následně provedeno vzájemné porovnání. Toto porovnání, za použití druhého nástroje, zde bylo provedeno pro celkem 4 funkce viz tabulka 7. Funkce byly zvoleny tak, aby každá patřila do jiné z kategorií dle tabulky 2. Pro každou skupinu běhů, 30 nejlepších versus 30 nejhorších, na dané účelové funkci došlo k jejich převodu na síť a vzájemné porovnání vlastností. Pro zpřehlednění jsou v této práci popsány jen vlastnosti, u kterých byly pozorovány významné odlišnosti mezi skupinou nejlepších a nejhorších běhů. Ostatní vygenerované histogramy a výsledky ohodnocení jsou přiloženy na médiu. Při porovnávání histogramů jsou vždy nejlepší běhy označeny jako skupina číslo 1 a nejhorší běhy jako skupina číslo 2.

Tabulka 6: Řídící parametry algoritmu DE zvolené pro generování logů k následné analýze

Parametr	Zvolená hodnota
Dimenze problému	10
Počet generací	3000
Verze algoritmu	DE/rand/1/bin
Účelové funkce	1 - 30
Počet běhů	500
Velikost populace	30
CR	0,9
F	0,5

8.1 Výsledky pro převod pozitivní interakce

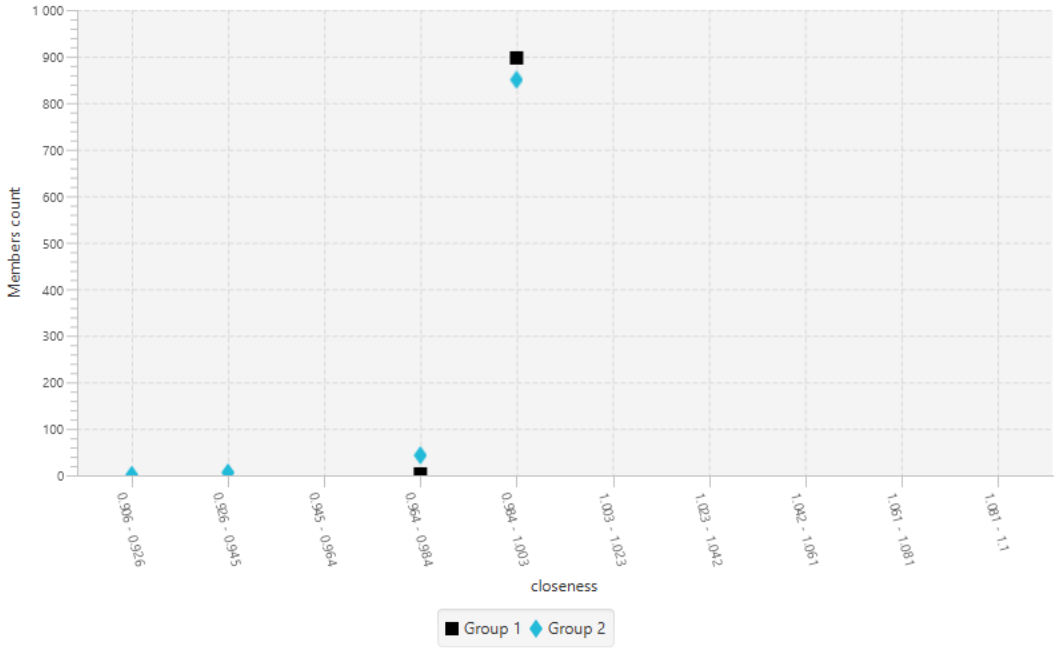
V tomto převodu jde o zachycení pozitivních interakcí mezi jedinci. Při vzájemném porovnávání dvou skupin se podstatně lišily pouze běhy uskutečněné s účelovou funkci číslo 10. U ostatních nebyly pozorovány rozdíly. Pravděpodobně z důvodu vytvoření kompletního grafu vykazujícího stejné hodnoty pro každou skupinu.

8.1.1 Účelová funkce č. 10

Na obrázku 16, lze vidět že pro účelovou funkci číslo 10, byl zaznamenán rozdíl hodnot CC mezi nejlepší a nejhorší skupinou běhů, kdy skupina 1 dosahovala v průměru vyšších hodnot než skupina 2. Znamená to tedy, že u skupiny lepších běhů jedinci interagovali s větším počtem jedinců. Tento fakt lze také vypozorovat na jednotlivých výstupních stupních vrcholů, kde u skupiny 2

Tabulka 7: Zvolené funkce pro analýzu

Kategorie	Číslo	Funkce
Unimodální funkce	1	Rotated High Conditioned Elliptic Function
Jednoduché multimodální funkce	10	Shifted Schwefel's Function
Hybridní funkce	20	Hybrid Function 4 (N=4)
Kompozitní funkce	25	Composition Function 3 (N=3)



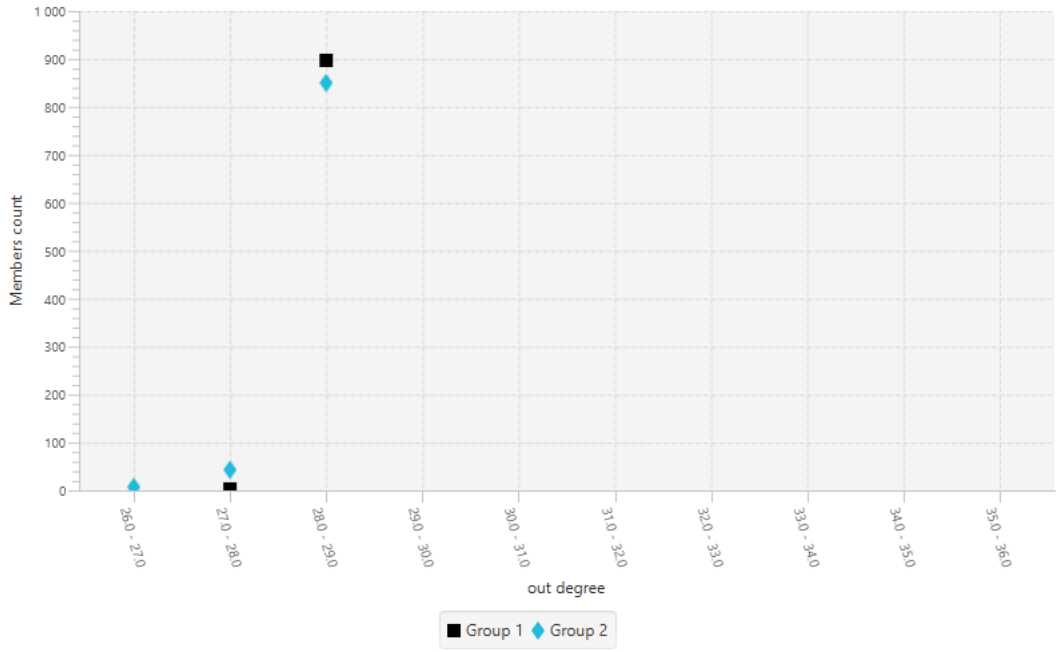
Obrázek 16: Porovnání CC pro funkci 10 - pozitivní interakce

došlo k výskytu 50 vrcholů se stupněm 26 - 28 , zatímco u druhé skupiny byly v tomto rozmezí vrcholy pouze 3, viz obrázek 17. Tyto zjištění byla reflektována i v dalších porovnávaných vlastnostech. Například hodnoty BC se u skupiny lepších běhů držely v rozmezí 0 - 0,107 pro všechny vrcholy, z čehož vyplývá, že většina předaného genomu nově vzniklému jedinci byla přímo mezi jedinci, bez prostředníka.

8.2 Výsledky pro převod vážená pozitivní interakce

V tomto případě bylo porovnání zajímavější oproti předchozímu typu převodu. Vzájemné rozdíly mezi nejlepšími a nejhoršími se projevily u všech čtyř funkcí. Avšak při vzájemném porovnání napříč funkcemi nebyla vysledována žádná závislost mezi hodnotami první a druhé skupiny, která by se projevila u všech čtyř funkcí.

Nejčastěji se lišily hodnoty CC, kdy u funkcí 1 a 20 nabývala skupina číslo jedna nižších hodnot, zatímco u zbylých dvou funkcí tomu bylo naopak. Stejné rozložení hodnot výstupní síly vrcholu bylo pozorováno u funkcí číslo 1, 20 a 25, kdy skupina první nabývala průměrně



Obrázek 17: Porovnání výstupního stupně vrcholů pro funkci 10 - pozitivní interakce

vyšší hodnoty než skupina druhá. U funkce číslo 10 se jako u jediné častěji vyskytovaly vrcholy s nižším stupněm, kdy větší počet se nacházel u skupiny číslo dvě.

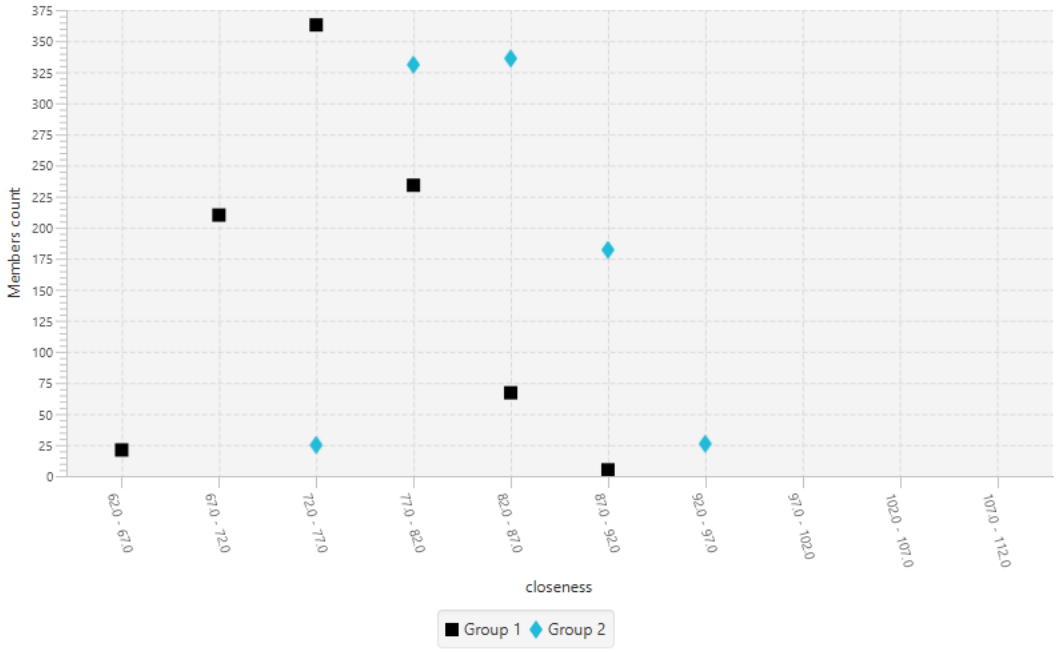
8.2.1 Účelová funkce č.1

Výrazný rozdíl byl zde pozorován u vlastnosti CC, kdy průměrná hodnota u skupiny jedna je 75, zatím co u druhá skupina dosahovala v průměru hodnoty 84, jak je vidět na obrázku 18. Což znamená, že u druhé skupiny docházelo častěji k interakcím s vylepšením jedince.

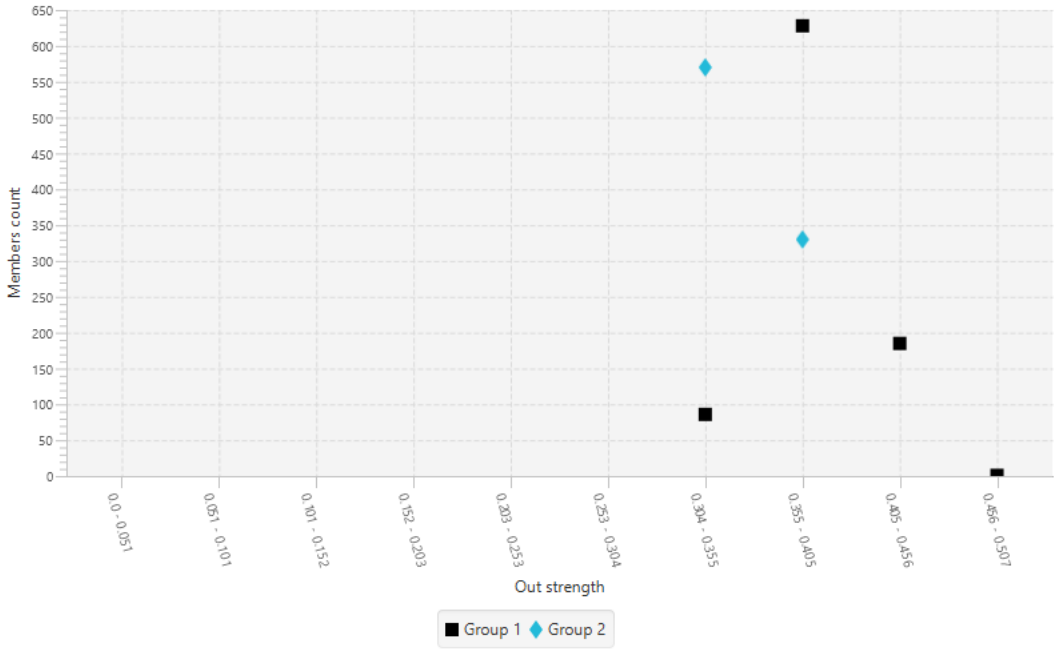
Další rozdíl je možno pozorovat na obrázku 19, kde výstupní síla jednotlivých vrcholů ve skupině 1 dosahovala mírně vyšších hodnot, v průměru 0,39. U druhé skupiny byl průměr výstupní síly vrcholu 0,35. Z toho vyplývá, že interakce u druhé skupiny byly častější, tím pádem měly hrany nižší hodnoty což se promítlo do průměrného síly vrcholu.

8.2.2 Účelová funkce č.10

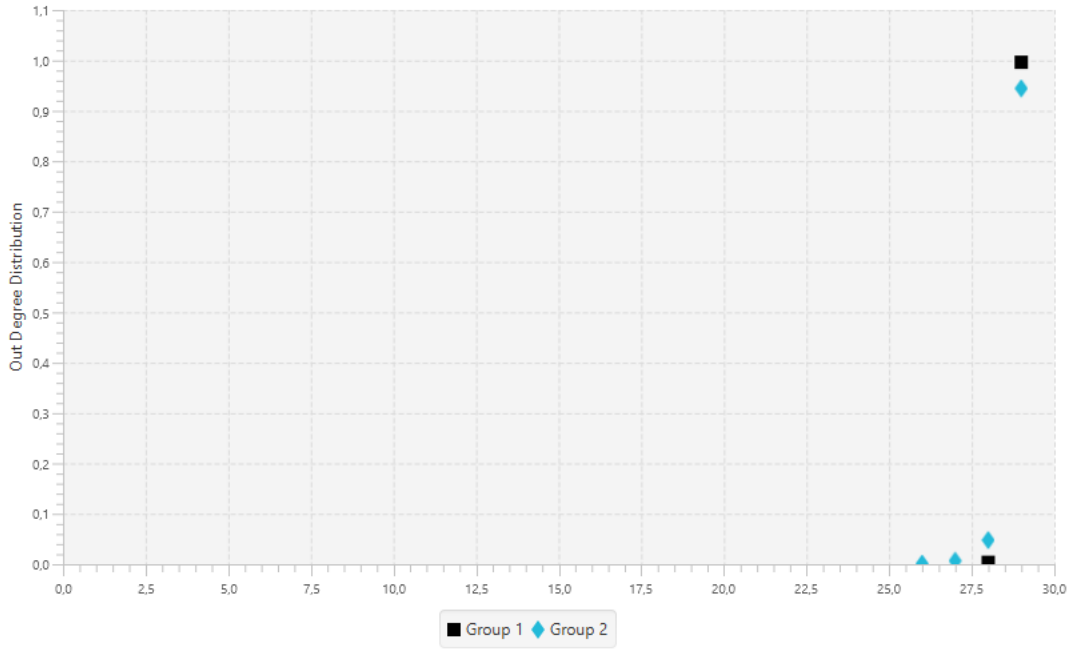
Pro tuto účelovou funkci lze pozorovat, že se zde vyskytovalo více vrcholů s nižším stupněm ve skupině č. 2. To znamená, že u vrcholů s nižším stupněm, došlo k vzájemné interakci s méně jedinci, kde došlo k vylepšení jedince. Tento jev lze pozorovat také u distribuce jednotlivých stupňů, na obrázku 20 pro výstupní stupeň vrcholu. Další rozdíl lze také pozorovat u BC, na obrázku 21 je vidět, že skupina 1 dosahuje nižších hodnot, průměrně 2,1, oproti skupině číslo dvě, kde průměrná hodnota BC pro vrchol je 2,7.



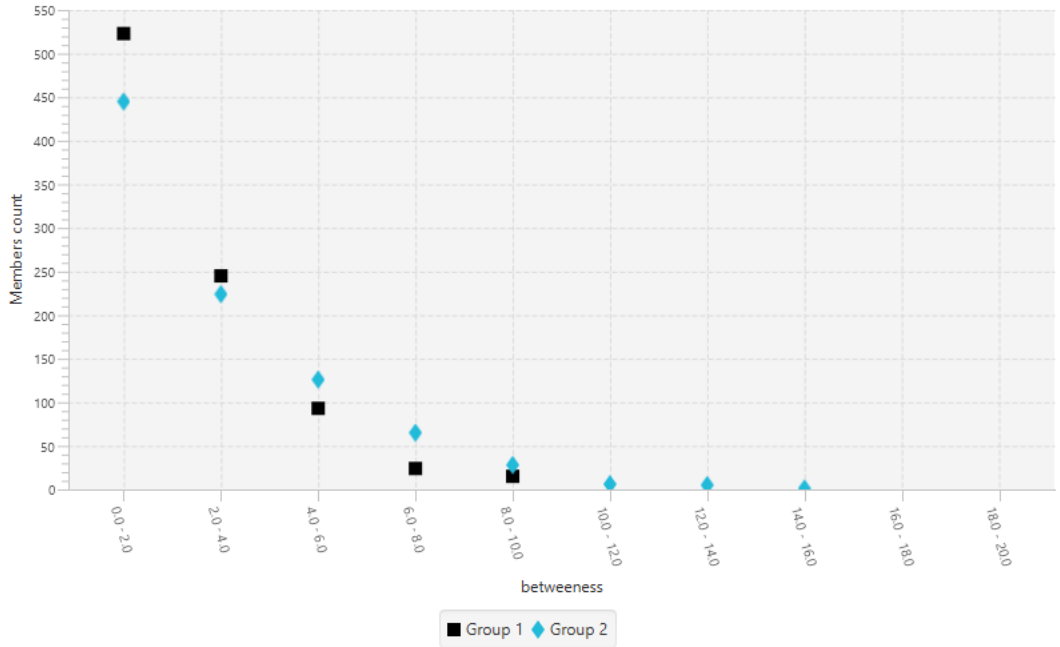
Obrázek 18: Porovnání CC pro funkci 1 - pozitivní vážená interakce



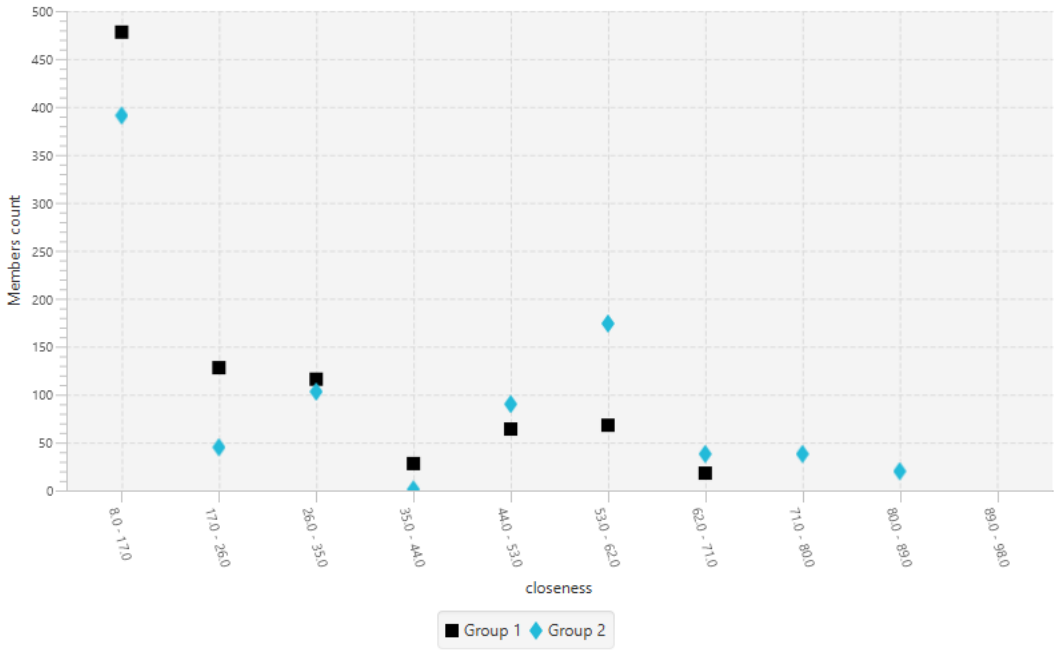
Obrázek 19: Porovnání výstupní síly vrcholu pro funkci 1 - pozitivní vážená interakce



Obrázek 20: Porovnání distribuce výstupního stupně vrcholu pro funkci 10 - pozitivní vážená interakce



Obrázek 21: Porovnání hodnot BC pro funkci 10 - pozitivní vážená interakce



Obrázek 22: Porovnání hodnot CC pro funkci 20 - pozitivní vážená interakce

8.2.3 Účelová funkce č.20

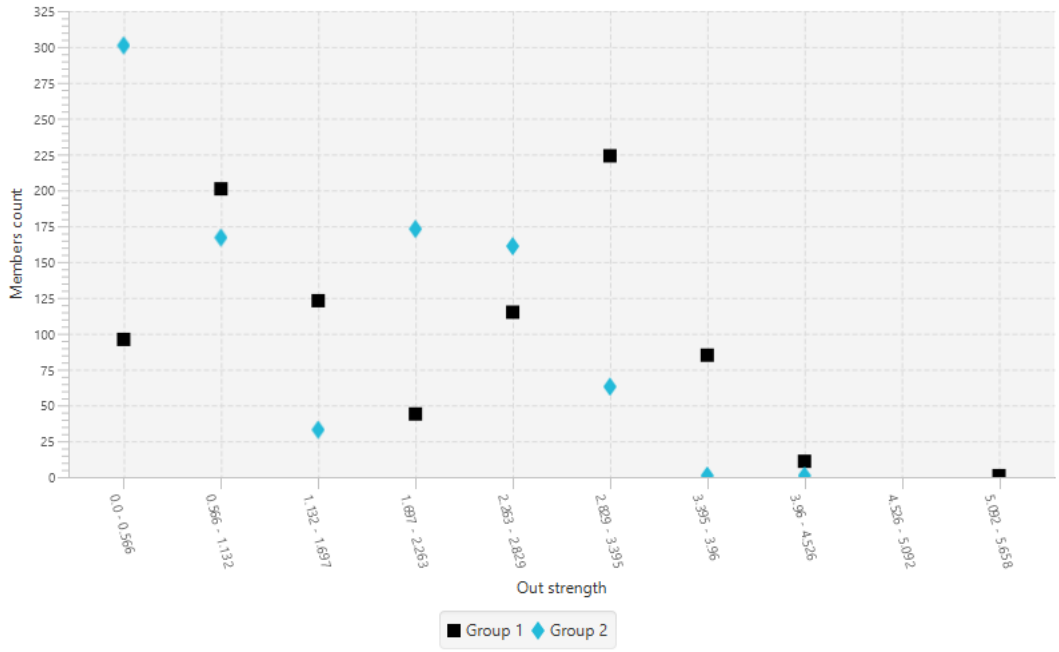
U tohoto způsobu převodu pro běhy s účelovou funkcí č. 20 se vyskytly pozorovatelné rozdíly u hodnot CC. První skupina dosahovala v průměru hodnoty 23, zatímco u druhé skupiny byla průměrná hodnota CC vyšší a to 34. Tuto situaci lze vidět na obrázku 22. Vyšších hodnot nabývala také hodnota výstupní síly vrcholu v první skupině, konkrétně 2,04 oproti hodnotě 1,4 pro skupinu číslo dva, zobrazeno na obrázku 23. Toto rozložení hodnot pro CC a výstupní sílu vrcholu v rámci skupin, bylo pro tento typ převodu možné pozorovat i na účelové funkci číslo 1.

8.2.4 Účelová funkce č.25

Při vzájemném porovnání na této účelové funkci byl výraznější rozdíl pozorován pouze u hodnoty CC, kdy vrcholy první skupiny nabývaly v průměru hodnoty 16,5. Průměrná hodnota vrcholů v druhé skupině byla 14,8. Mírný rozdíl se nacházel i u průměrné hodnoty výstupní síly vrcholu. Kdy první skupina nabývala průměrné hodnoty 2,752 oproti hodnotě 2,746 u druhé skupiny.

8.3 Výsledky pro převod negativní interakce

Při analýze a vzájemnému porovnání dvou skupin běhů, kde byla pro převod na síť využita metoda převodu zaznamenávající negativní interakce, nebyly pro 4 vybrané účelové funkce zjevné rozdíly mezi lepší a horší skupinou. Je velmi pravděpodobné, že tento fakt je způsoben tím, že alespoň jednou, pro každou vzájemnou interakci mezi jakoukoliv dvojicí jedinců, nedošlo ke vzniku nového lepšího jedince. Tento postup byl stejný pro běh na kterékoliv z vybraných



Obrázek 23: Porovnání výstupního síly vrcholu pro funkci 20 - pozitivní vážená interakce

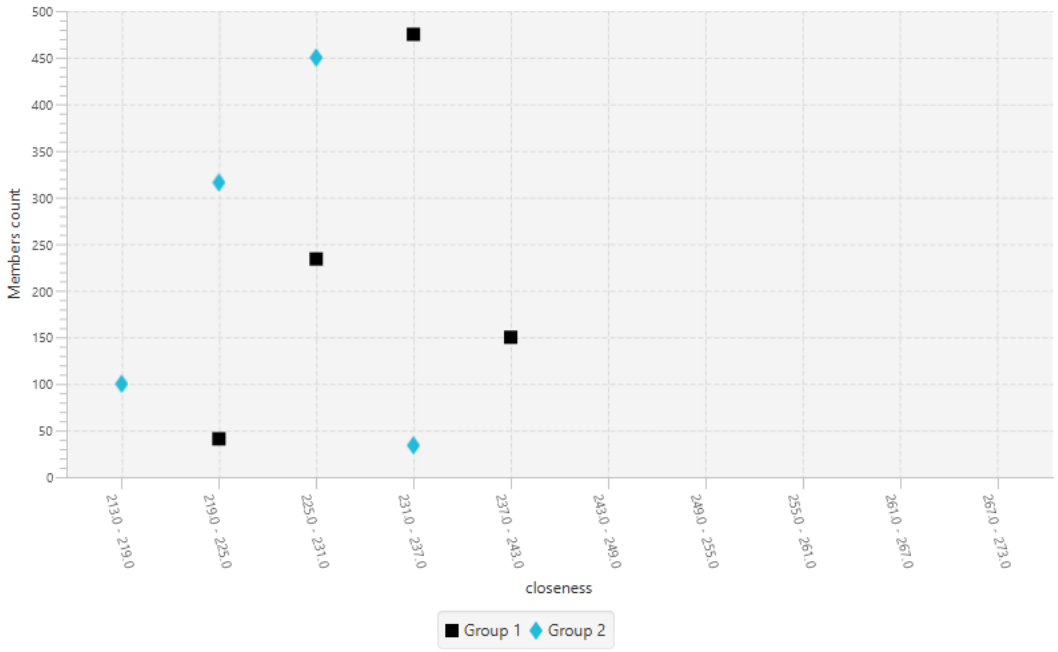
účelových funkcí, vzniklá síť byla tedy vždy stejná, jednalo se o úplný graf, a tedy porovnání nevykazovalo vzájemné rozdíly pro jednotlivé skupiny.

8.4 Výsledky pro převod vážená negativní interakce

Jak je vidět z tabulky 8, u tohoto převodu byly zaznamenány vlastnosti, u kterých byl na všech funkcích pozorován stejný trend v rozložení mezi skupinou nejlepších a nejhorších běhů. První z těchto vlastností byla CC. Pro tuto vlastnost byla u všech funkcí průměrná hodnota první skupiny vždy vyšší. Čím vyšší CC vrcholu, tím nižší je vzdálenost z tohoto vrcholu do ostatních. V rámci tohoto převodu to znamená, že ve skupině nejlepších běhů docházelo k častějším interakcím, kdy nebyl vytvořen nový jedinec, protože čím nižší jsou vzdálenosti mezi vrcholy tím k více interakcím došlo. Toto potvrzuje i druhá vlastnost, kdy průměrná hodnota výstupní síly vrcholu byla u první skupiny běhů nižší, rovněž pro všechny funkce.

Tabulka 8: Výsledky pro převod vážená negativní interakce

Metrika	Funkce 1		Funkce 10		Funkce 20		Funkce 25	
	skupina 1	skupina 2	skupina 1	skupina 2	skupina 1	skupina 2	skupina 1	skupina 2
CC	233	225	116	94	245	224	70	60
Výstupní síla vrcholu	0,125	0,13	0,27	0,34	0,12	0,14	0,46	0,57



Obrázek 24: Porovnání hodnot CC pro funkci 1 - negativní vážená interakce

8.4.1 Účelová funkce č.1

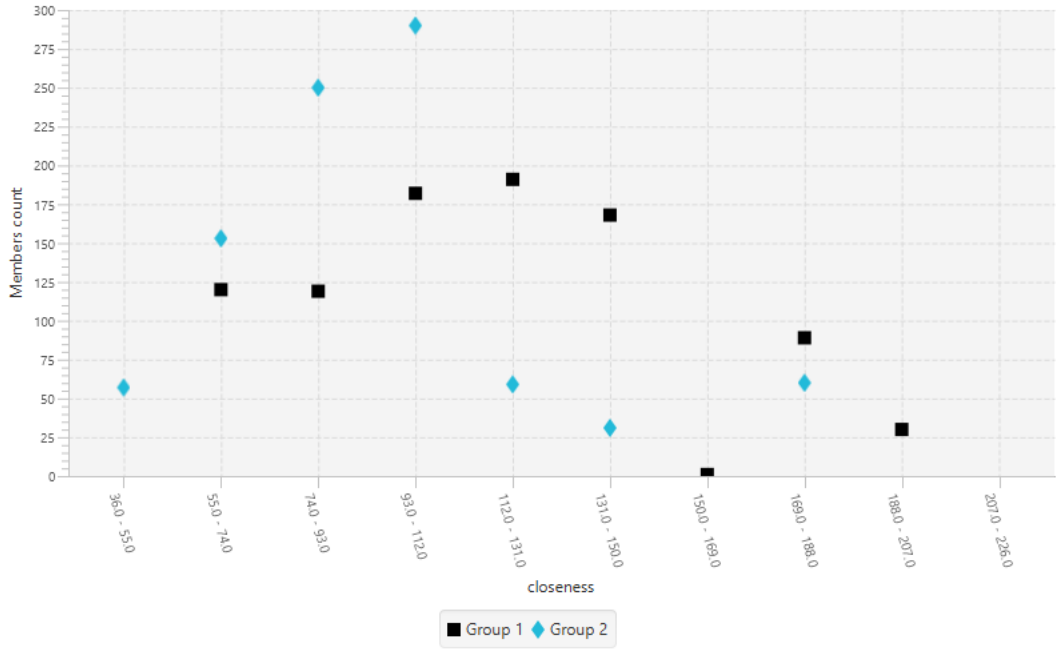
U zkoumání rozdílů mezi dvěma skupinami běhů pro tuto účelovou funkci, byla pozorována opačná tendence než pro váženou pozitivní interakci na stejné funkci. Hodnoty CC byly v průměru vyšší u první skupiny, kdy zde vrcholy nabývaly průměrné hodnoty 233. Ve druhé skupině byla průměrná hodnota 225, tyto hodnoty lze porovnat na obrázku 24.

8.4.2 Účelová funkce č.10

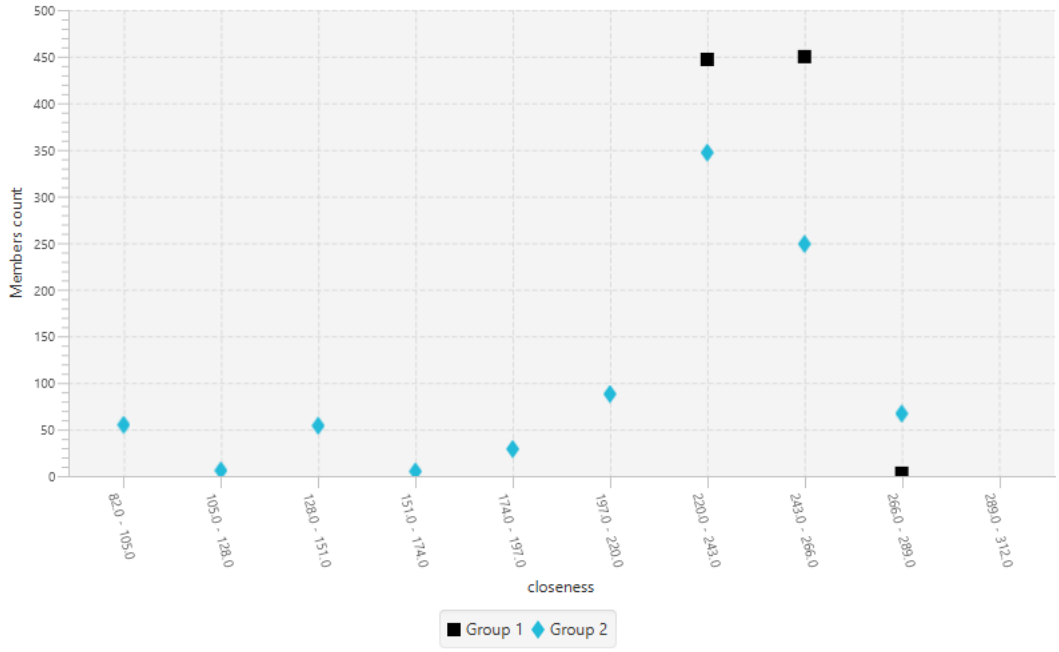
Stejné tendence jako u předchozí zmiňované funkce reflektovala i tato. Opět byl hlavní rozdíl pozorován na vlastnosti CC, kde skupina lepších běhů dosahovala v průměru vyšších hodnot, jak je vidět na obrázku 25.

8.4.3 Účelová funkce č.20

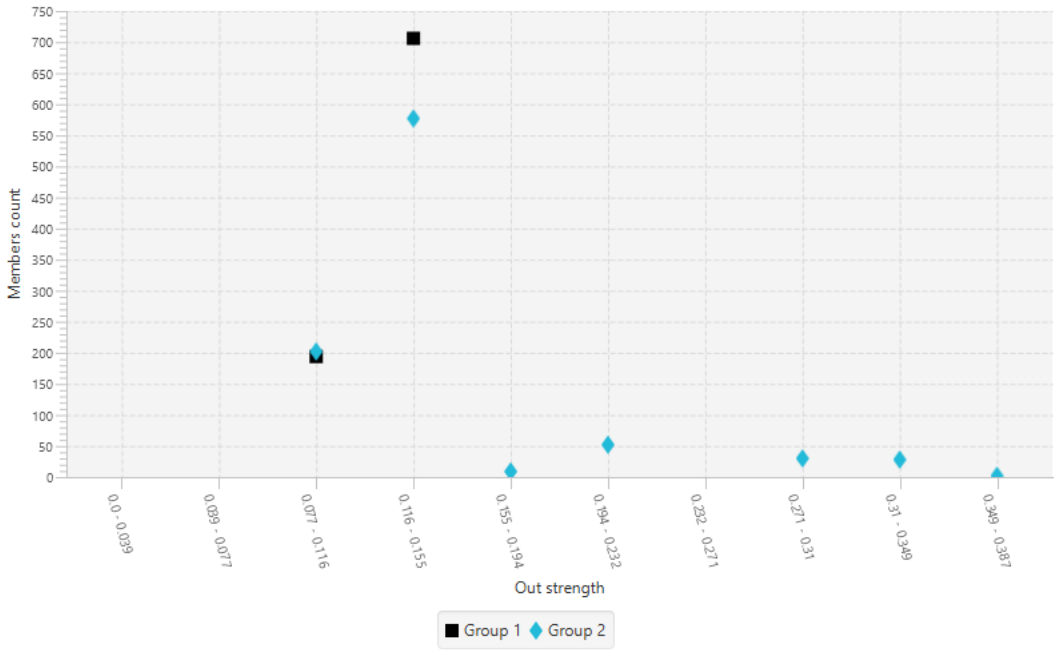
I u této funkce bylo vypozorováno, že skupina lepších běhů dosahuje u tohoto způsobu převodu vyšší hodnotu CC, v průměru 245, oproti skupině druhé složené z běhů nejhorších, kde vrcholy dosahovaly v průměru hodnoty 224. Tato vlastnost je vyobrazena na obrázku 26. Z obrázku 27 vyplývá, že ve skupině 2 má více vrcholů větší výstupní sílu. Průměrná výstupní síla vrcholu první skupiny je 0,12, zatím co průměr v druhé skupině je 0,14.



Obrázek 25: Porovnání hodnot CC pro funkci 10 - negativní vážená interakce



Obrázek 26: Porovnání hodnot CC pro funkci 20 - negativní vážená interakce



Obrázek 27: Porovnání hodnot výstupní síly vrcholu pro funkci 20 - negativní vážená interakce

8.4.4 Účelová funkce č.25

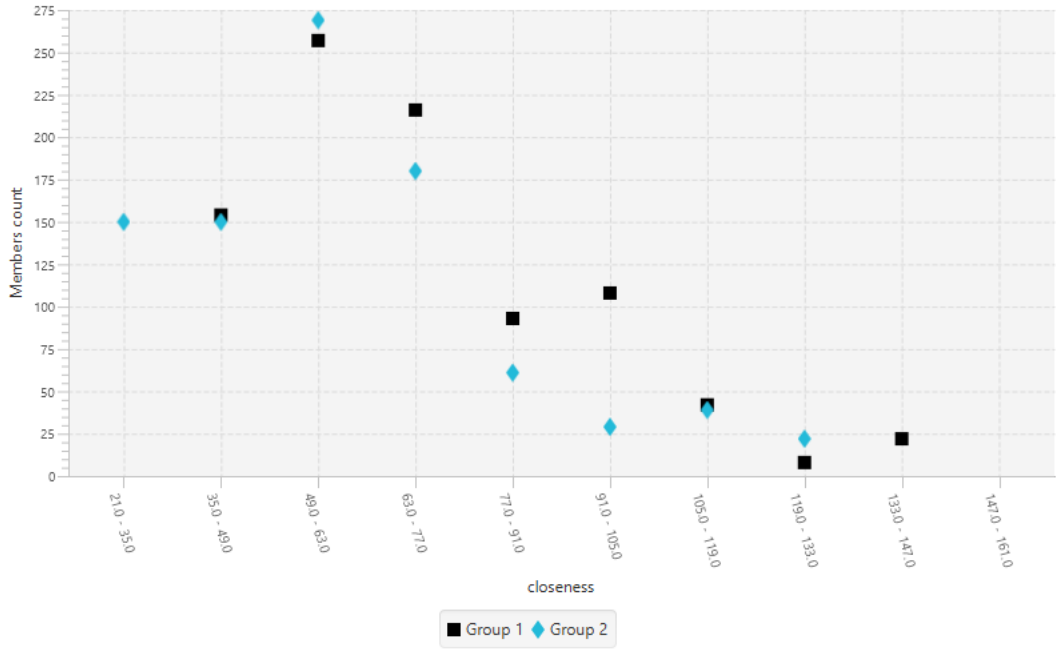
Ani poslední z funkcí v tomto typu převodu nenarušila zde sledovaný poměr mezi skupinami, kdy první skupina dosahuje v průměru vyšších hodnot CC, v tomto případě má konkrétně první skupina průměr 70 a druhá skupina dosahuje průměrné hodnoty 60. Jednotlivé ohodnocení vrcholů lze pozorovat na obrázku 28.

8.5 Výsledky pro převod pozitivní vážená interakce s prvky mravenčí optimalizace

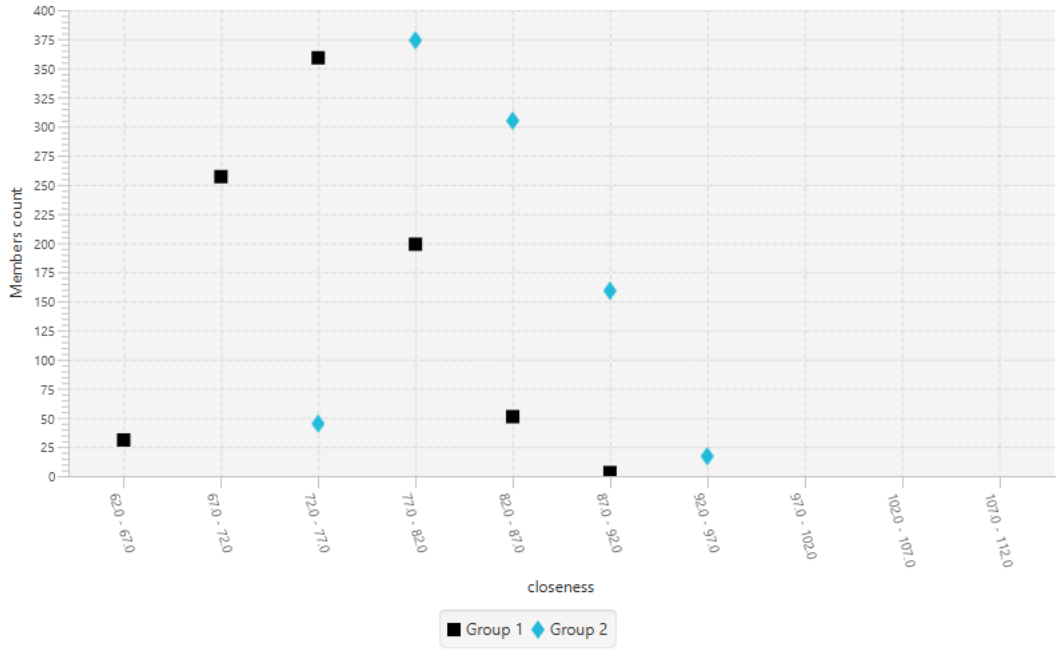
Hlavním rozdíl mezi skupinami tento převod vykazoval na vlastnosti CC, kde vesměs reflektoval výsledky popsané u převodu pozitivní vážené interakce. Pro funkce 1 a 20 dosahovala první skupina běhů průměrné hodnoty CC nižší než skupina druhá. U funkcí 10 a 25 tomu bylo opačně, a první skupina dosahovala průměrných hodnot CC vyšších. Přesné dosažené hodnoty jsou zmíněny níže, u popisu pro každou funkci.

8.5.1 Účelová funkce č.1

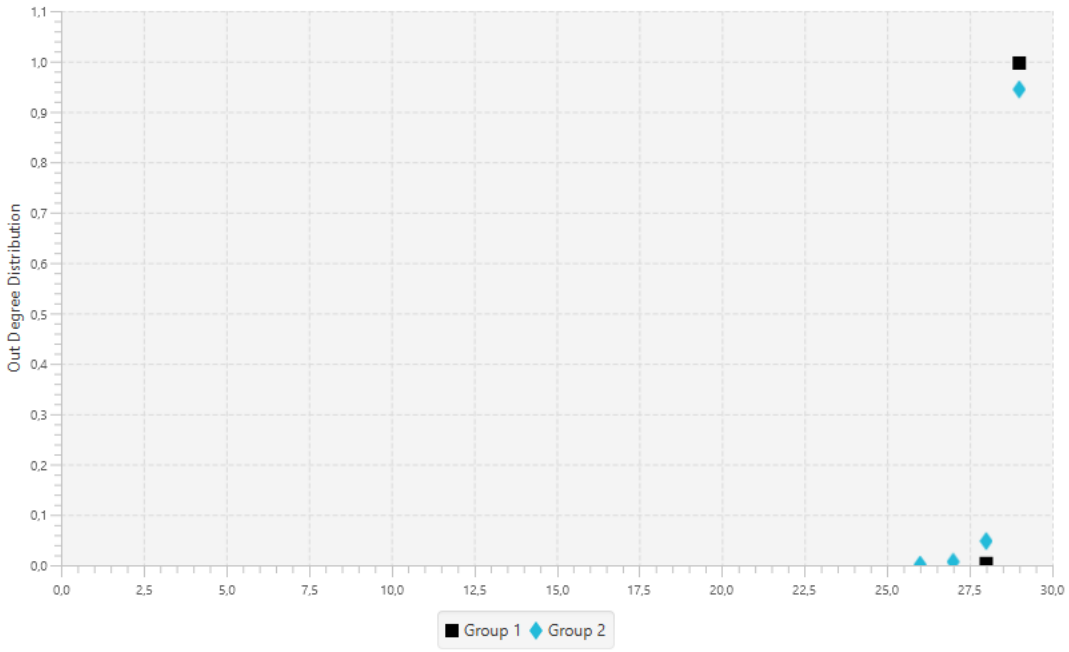
Na většině vlastností zde nebyl pozorován rozdíl mezi nejlepšími a nejhoršími běhy. Pouze u CC zde došlo k odlišnostem, kdy skupina 1 dosahovala průměrné hodnoty 75. Skupina 2 měla v průměru hodnoty CC u vrcholů vyšší a to s průměrnou hodnotou 83. Počet vrcholů v určitém rozmezí hodnot CC lze vidět na obrázku 29.



Obrázek 28: Porovnání hodnot CC pro funkci 25 - negativní vážená interakce



Obrázek 29: Porovnání hodnot CC pro funkci 1 - pozitivní vážená interakce s prvky mravenčí optimalizace



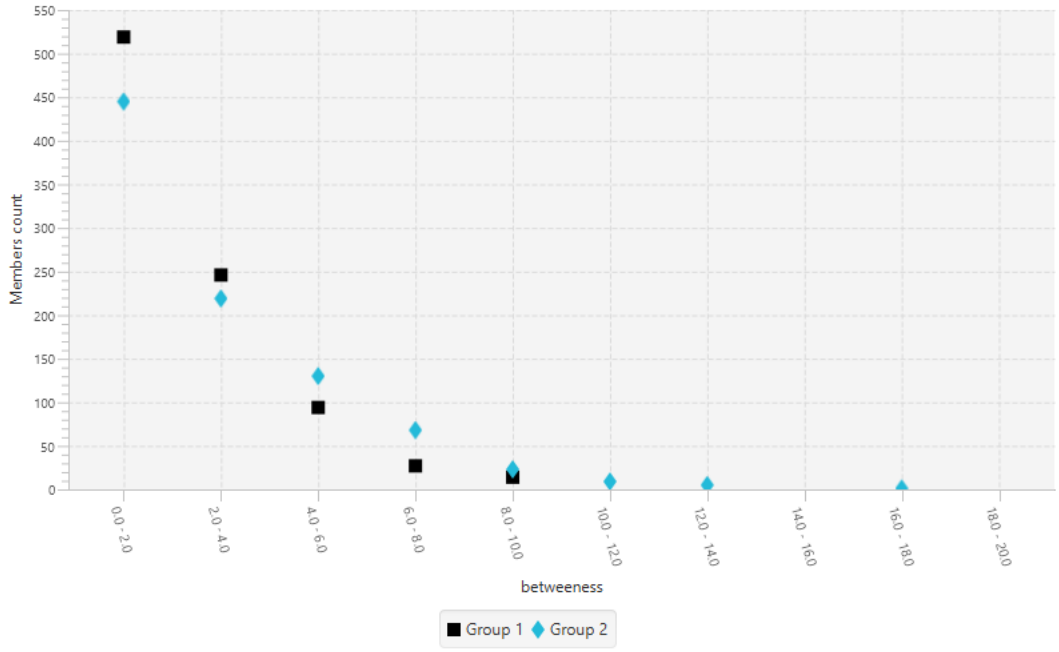
Obrázek 30: Porovnání distribuce výstupních stupňů pro funkci 10 - pozitivní vážená interakce s prvky mravenčí optimalizace

8.5.2 Účelová funkce č.10

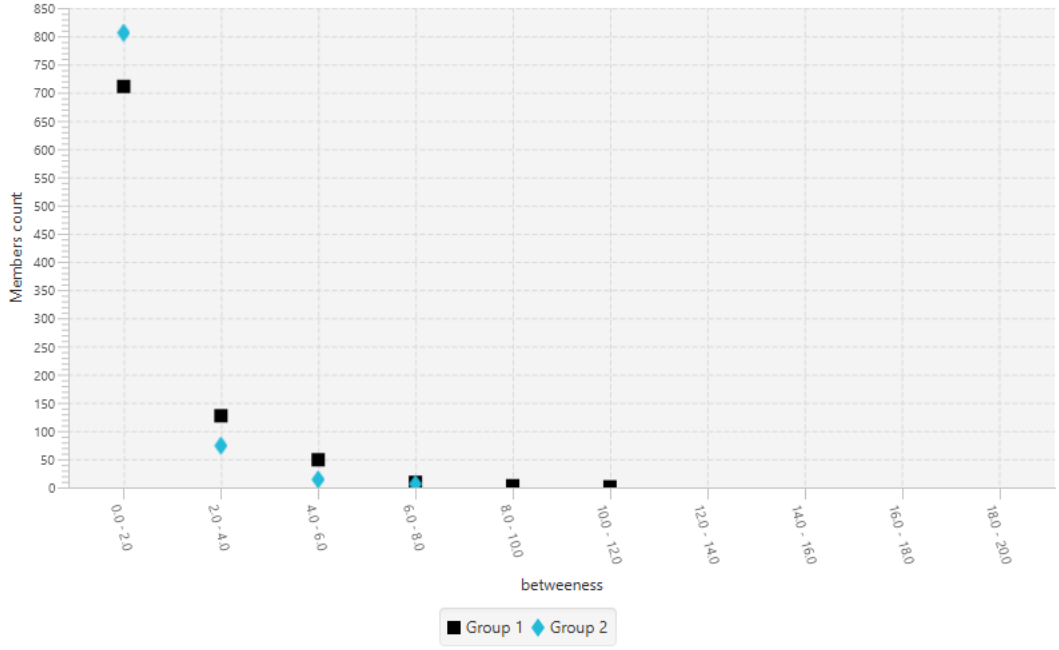
U této funkce ze rozdíl mezi skupinami dal pozorovat na rozložení jednotlivých výstupních stupňů vrcholů, u druhé skupiny se nacházelo více jedinců, u kterých nedošlo ani jednou k interakci s více než jedním jedincem. To znamená, že se zde nacházely vrcholy s výstupním stupněm 26 a 27, zatímco v první skupině byl nejnižší výstupní stupeň vrcholu 28. Rozložení výstupních stupňů je zobrazeno na obrázku 30. Dalším rozdílem, který lze pozorovat u této účelové funkce je rozdíl u průměrných hodnot BC, kde druhá skupina dosahovala v průměru vyšší hodnoty než první, 2,8 oproti 2,1. Zobrazeno na obrázku 31. U průměrné hodnoty CC v tomto případě mírně vyšších hodnot dosahovala první skupina.

8.5.3 Účelová funkce č.20

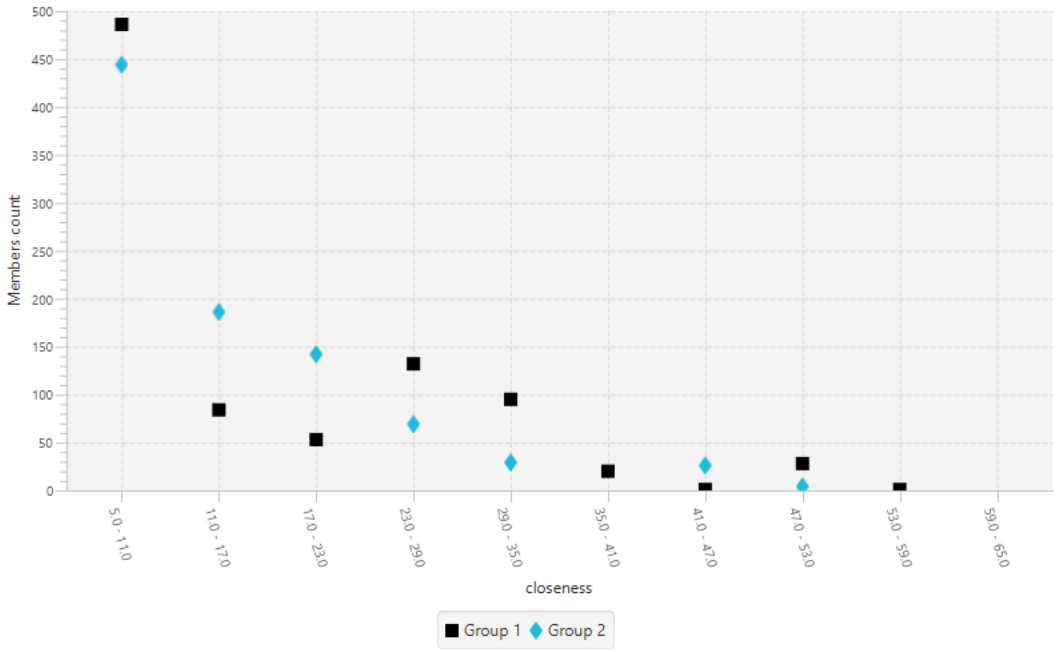
U této funkce dosahovala průměrná hodnota BC první skupiny téměř dvojnásobku průměrné hodnoty druhé skupiny. První skupina měla průměrnou hodnotu vyšší a to 1,2 přičemž druhá skupina dosahovala průměrné hodnoty 0,7. Rozložení hodnot BC lze vidět na obrázku 32. Hodnoty CC pro tuto funkci následovaly trend pozorovaný již u stejného převodu na funkci č.1, kdy první skupina dosahuje v průměru nižších hodnot než skupina druhá. Konkrétně průměr hodnot je u první skupiny 22,5 a u druhé skupiny 33,7.



Obrázek 31: Porovnání hodnot BC pro funkci 10 - pozitivní vážená interakce s prvky mravenčí optimalizace



Obrázek 32: Porovnání hodnot BC pro funkci 20 - pozitivní vážená interakce s prvky mravenčí optimalizace



Obrázek 33: Porovnání hodnot CC pro funkci 25 - pozitivní vážená interakce s prvky mravenčí optimalizace

8.5.4 Účelová funkce č.25

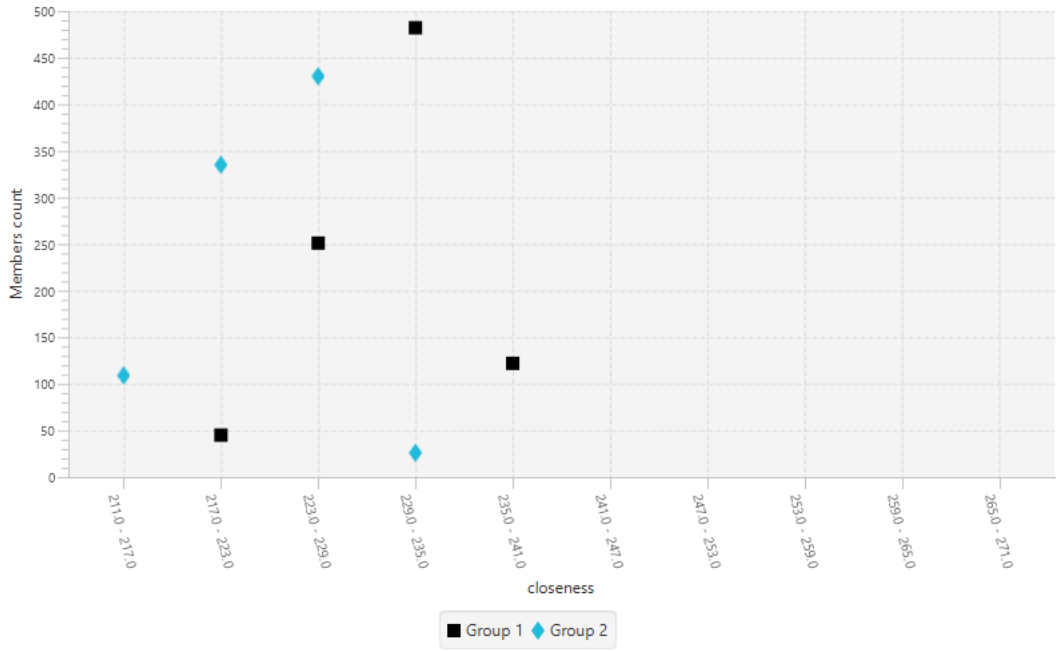
Pro tuto funkci byl největší rozdíl při porovnání pozorován u hodnot CC, kdy první skupina dosahovala mírně vyšších hodnot s průměrem 16,5. Druhá skupina nabývala hodnot v průměru 14,8. Porovnání hodnot CC je zobrazeno na obrázku 33.

8.6 Výsledky pro převod negativní vážená interakce s prvky mravenčí optimalizace

V rámci tohoto převodu bylo pozorováno stejné rozložení hodnot v rámci vlastnosti CC. Není velkým překvapením, že se toto rozložení moc neliší od porovnání v rámci převodu negativní vážená interakce. Opět zde u všech zkoumaných funkcí dosahovaly běhy z první skupiny vyšších hodnot než u skupiny dvě a to u všech čtyř funkcí.

8.6.1 Účelová funkce č.1

Pro tuto účelovou funkci lze pozorovat rozdíl u hodnoty CC, obrázek 34, kdy první skupina dosahuje v průměru hodnoty 231. Oproti tomu průměrná hodnota CC pro druhou skupinu je nižší, konkrétně vrcholy v této skupině dosahovaly průměrné hodnoty 223.



Obrázek 34: Porovnání hodnot CC pro funkci 1 - negativní vážená interakce s prvky mravenčí optimalizace

8.6.2 Účelová funkce č.10

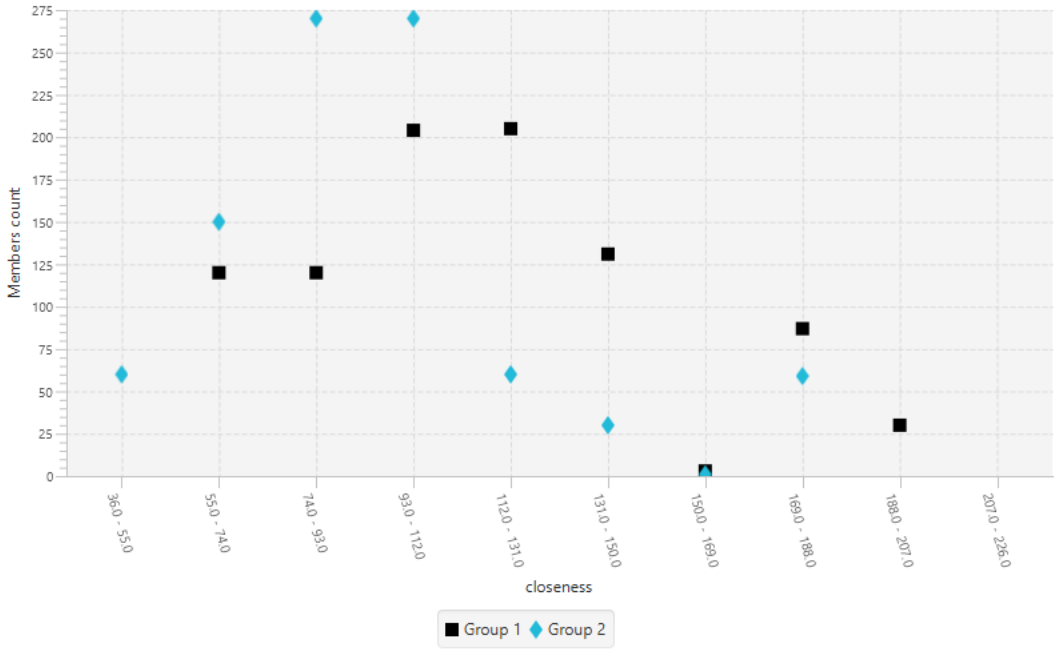
Podobně jako u předchozí funkce i zde skupina obsahující lepší běhy dosahovala vyšších hodnot CC než skupina druhá. Konkrétně vrcholy první skupiny dosahovaly průměrné hodnoty CC 116. Průměrná hodnota vrcholu druhé skupiny byla 94. Výskyt jednotlivých hodnot lze vidět na obrázku 35.

8.6.3 Účelová funkce č.20

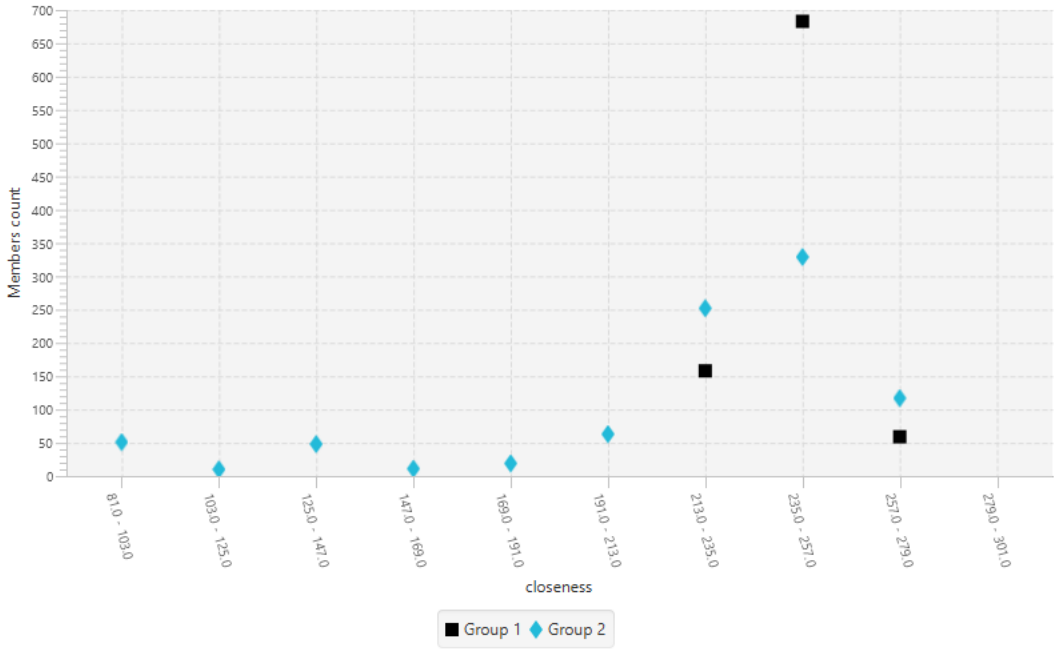
I tato v pořadí další funkce pro tento převod zůstala u stejně rozložených hodnot pro CC. Skupina jedna dosahovala průměrné hodnoty vrcholu 242. Ve druhé skupině byla průměrná dosažená hodnota vrcholu 222. Z obrázku 36 je tedy patrné, že stejně jako v předchozích případech skupina lepších běhů dosahuje vyšší hodnoty.

8.6.4 Účelová funkce č.25

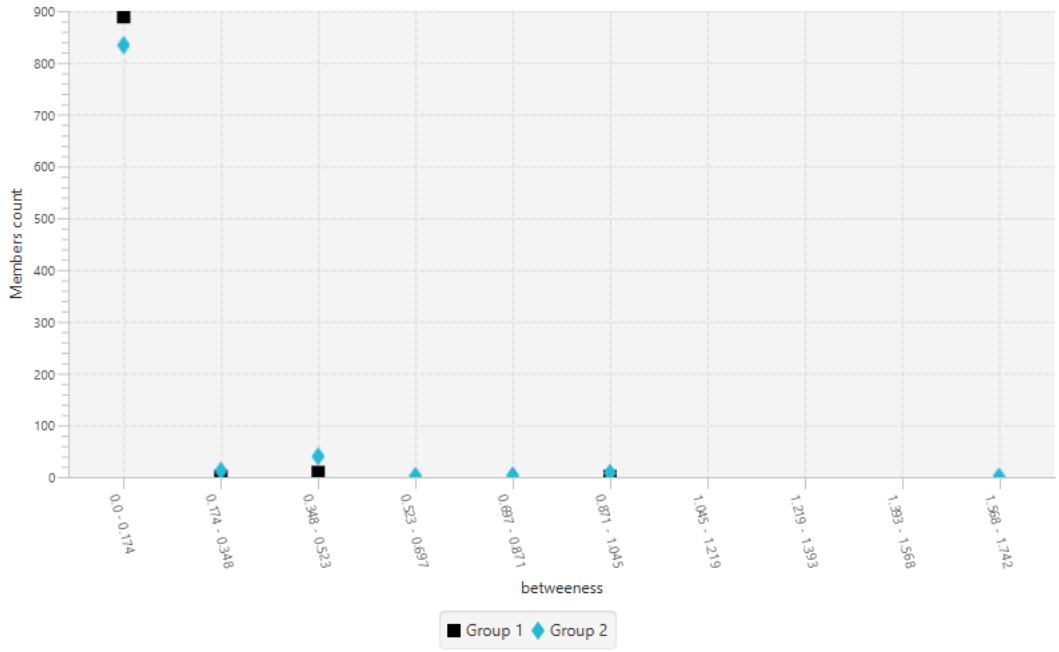
Pro tuto funkci, jak je vidět na obrázku 37, byl patrný rozdíl mezi skupinami u dosahovaných hodnot BC. První skupina dosahovala v průměru nižších hodnot, konkrétně průměrné hodnoty vrcholu 0,0065. Zatímco vrcholy v druhé skupině dosahovaly průměrně hodnoty 0,04. Průměrná hodnota druhé skupiny byla téměř 6krát vyšší něž u první skupiny. Čím vyšší je hodnota BC, tím více informací skrz daný vrchol prochází. V našem kontextu se tedy ve druhé skupině nacházelo více vrcholů, přes které docházelo k častějšímu šíření genomu.



Obrázek 35: Porovnání hodnot CC pro funkci 10 - negativní vážená interakce s prvky mravenčí optimalizace



Obrázek 36: Porovnání hodnot CC pro funkci 20 - negativní vážená interakce s prvky mravenčí optimalizace



Obrázek 37: Porovnání hodnot BC pro funkci 25 - negativní vážená interakce s prvky mravenčí optimalizace

Při porovnání hodnoty CC zůstaly zachované tendence popisované u výše zmíněných funkcí. První skupina dosahovala hodnoty CC v průměru 69, druhá skupina hodnoty 59.

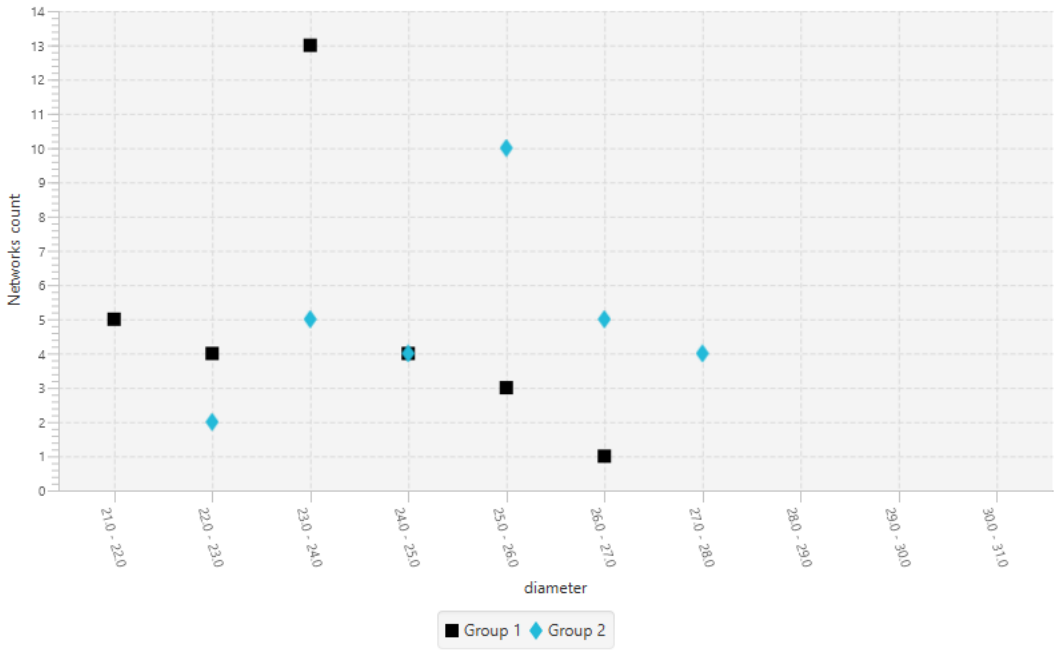
8.7 Výsledky pro převod nejlepší z generace

Při porovnání skupin u převodu zaznamenávajícího vznik nejlepšího jedince z generace nebyla zjištěna tendence pro vzájemné porovnání dvou skupin u všech funkcí. Avšak například nižších hodnot BC dosahovala první skupina u funkcí 1, 10 i 20, zatímco u funkce 25 tomu bylo opačně. Z toho vyplývá, že u funkcí 1,10 a 20 docházelo k předávání genomu mezi vrcholy spíš přímo, a u funkce 25 tomu bylo spíše přes prostředníky.

Druhým pozorovaným rozdílem byl počet vrcholů v jedné či druhé skupině. Pro funkce 1 a 20 vykazovala první skupina celkový počet vrcholů menší než skupina druhá. To znamená, že v těchto případech byl aktuální nejlepší jedinec z populace méně-krát vylepšen, a přesto tyto skupiny ve výsledku našli lepší řešení pro danou účelovou funkci.

8.7.1 Účelová funkce č.1

Rozdíl mezi jednotlivými skupinami u této funkce se nalézal i v celkovém počtu vrcholů. V první skupině se nacházelo celkem 43 141 vrcholů, zatímco ve druhé skupině se vyskytovalo 46 862 vrcholů.



Obrázek 38: Porovnání hodnot průměr grafu pro funkci 1 - nejlepší z generace

Zajímavý je také pohled na vlastnost průměr grafu, obrázek 38. Sítě v první skupině dosahovaly průměrné hodnoty 23,9, zatímco průměrná hodnota ve skupině druhé byla 25,8. Průměr grafu značí nejdelší možnou cestu genomu, který byl použit k vylepšení.

Pozorovaný rozdíl mezi skupinami byl i u hodnot BC. První skupina nabývala v průměru nižších hodnot, konkrétně 5 538, zatímco průměrná hodnota v druhé skupině byla 6 416.

8.7.2 Účelová funkce č.10

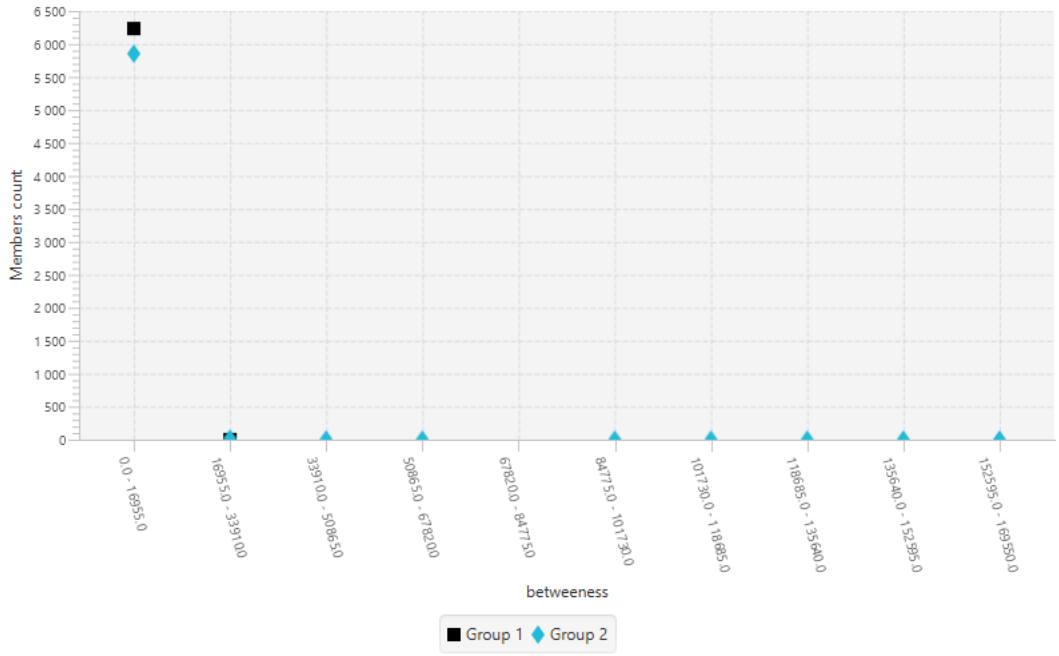
Počty vrcholů v jednotlivých skupinách zde byly následující, pro první skupinu to bylo 6 236 vrcholů a ve druhé skupině se nacházelo 5 878.

Rozdíl byl zpozorován i u rozložení hodnot BC, zobrazeno na obrázku 39. Pro první skupinu nabývaly vrcholy průměrné hodnoty 453. Vrcholy ve druhé skupině hodnoty 832.

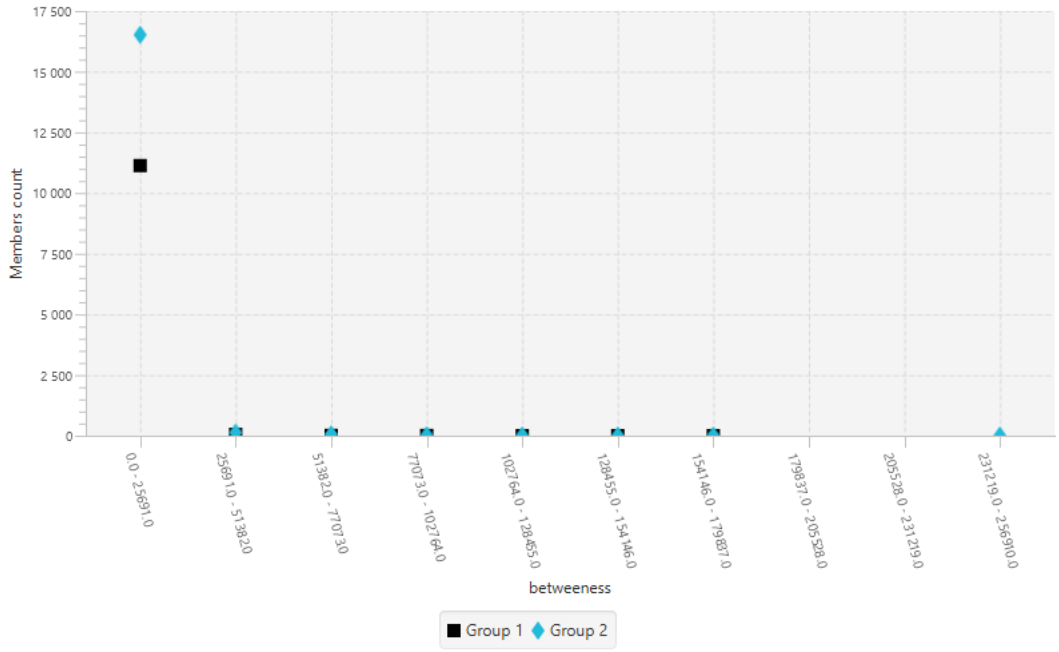
8.7.3 Účelová funkce č.20

Pro tuto funkci nabývala první skupina menšího počtu vrcholů než skupina druhá, stejně jako tomu bylo u funkce číslo 1. Konkrétní počty jsou 11 203 vrcholů v první skupině a 16 738 vrcholů ve skupině druhé.

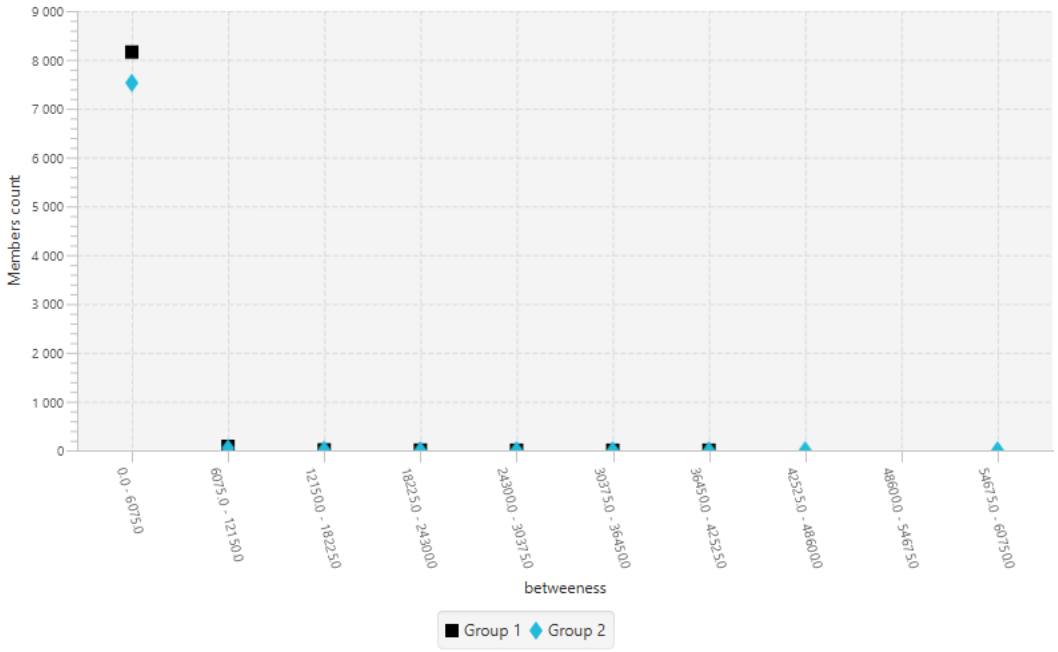
Další rozdíl byl pozorován u hodnoty BC, kdy vrcholy z první skupiny měly průměrnou hodnotu 1 439. U druhé skupiny to byla pro vrchol hodnota 2 319. Toto rozložení lze pozorovat na obrázku 40.



Obrázek 39: Porovnání hodnot BC pro funkci 10 - nejlepší z generace



Obrázek 40: Porovnání hodnot BC pro funkci 20 - nejlepší z generace



Obrázek 41: Porovnání hodnot BC pro funkci 25 - nejlepší z generace

8.7.4 Účelová funkce č.25

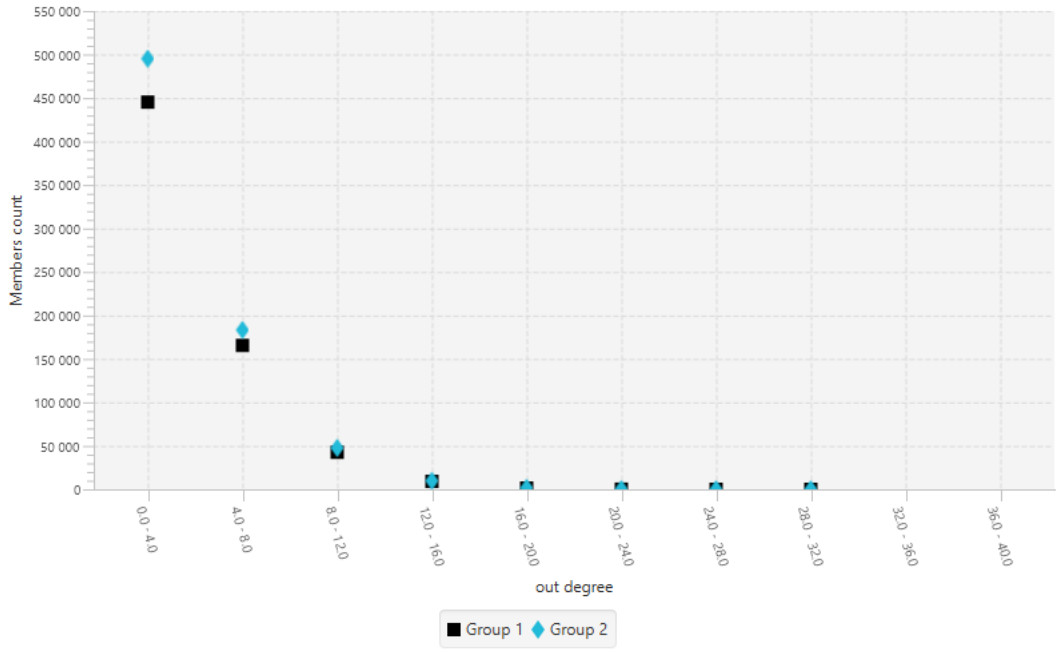
Pro tuto funkci došlo k vzájemnému rozdílu mezi skupinami u hodnot BC, vyobrazeno na obrázku 41. První skupina dosáhla v průměru hodnoty 729, zatímco druhá skupina hodnoty 574. Skupina jedna tedy měla průměrnou hodnotu BC vyšší. Jedná se tedy o opačný trend, než byl pozorován u všech tří předchozích skupin.

Rozdílný zde byl i počet vrcholů v jednotlivých skupinách. Počet vrcholů v první skupině byl vyšší oproti skupině druhé. V skupině jedna se nacházelo celkem 8 269 vrcholů a ve skupině dvě 7 596 vrcholů.

8.8 Výsledky pro postupný vývoj vrcholů

V kapitole u popisu jednotlivých převodů již byla naznačena paměťová náročnost tohoto převodu, proto zde byly vynechány pro porovnávání vlastnosti CC, BC, průměrná nejkratší délka cesty a průměr grafu. U všech těchto vlastností se počítá s cestami napříč grafem a zde nastává v rozsáhlých grafech problém.

Při tomto způsobu převodu je zajímavé pozorovat i celkové počty vrcholů v jednotlivých skupinách. Počty vrcholů nám mohou prozradit, ve které skupině docházelo k častějším vznikům nových vrcholů, což znamená, že zde častěji docházelo k vzájemným interakcím, kdy výsledkem byl vylepšený jedinec. Na funkcích 1 a 20 byl počet vrcholů menší u první skupiny. Zajímavé tedy je, že zde bylo během průběhu algoritmu DE vytvořeno méně vylepšených jedinců, i přesto však



Obrázek 42: Porovnání výstupních stupňů vrcholů pro funkci 1 - postupný vývoj vrcholů

tyto běhy dosahovaly lepších výsledků při hledání nejlepšího řešení na účelové funkci. U funkcí 10 a 25 tomu bylo naopak.

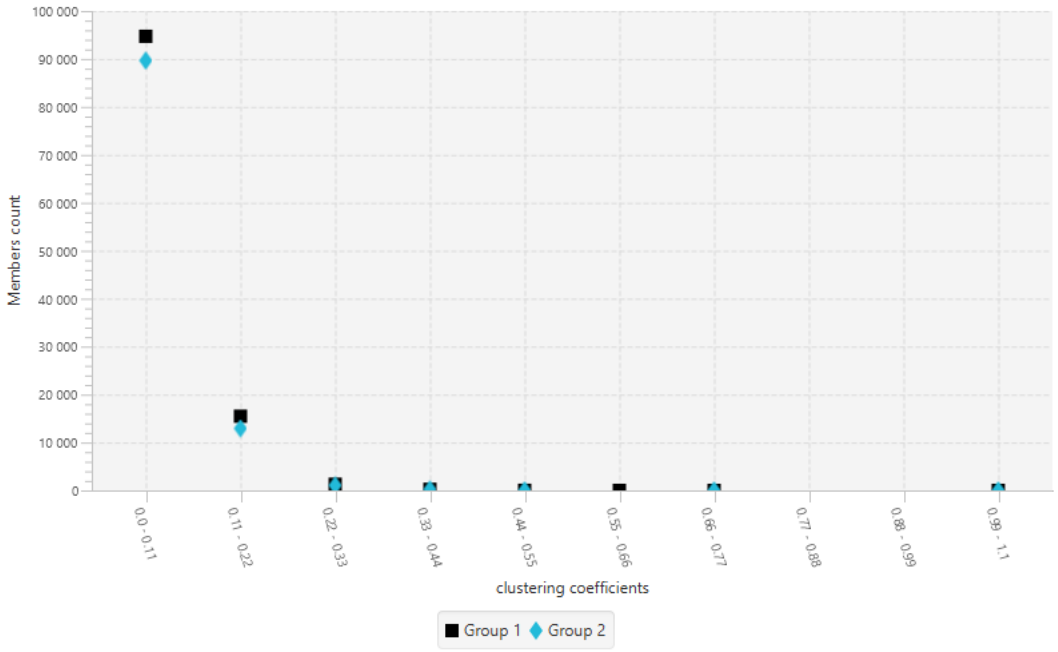
Druhou pozorovanou vlastností byl koeficient shlukování pro jednotlivé vrcholy. Vyšší průměrnou hodnotu koeficientu shlukování dosahovaly u funkcí 1,10 a 20 vrcholy v první skupině. Čím vyšší hodnota tohoto koeficientu u vrcholu, tím s větší částí vrcholů ze sítě došlo k interakci. V našem případě to tedy znamená, že u funkcí s vyšší průměrnou hodnotou koeficientu shlukování, každý jedinec interagoval s více jedinci z populace, než v případě nižší hodnoty.

8.8.1 Účelová funkce č.1

Pozoruhodný rozdíl byl mezi skupinami zaznamenán v celkovém počtu vrcholů, kdy v první skupině mělo 30 nejlepších běhů 664 191 vrcholů, zatímco ve druhé skupině byl celkový počet vrcholů 737 559. Z toho vyplývá, že v první skupině došlo ke vzniku menšího počtu vylepšených jedinců. Přibližné počty lze odečíst z obrázku 42, na kterém je zobrazeno porovnání výstupních stupňů vrcholů. Koeficient shlukování u první skupiny byl v průměru mírně vyšší než u skupiny druhé.

8.8.2 Účelová funkce č.10

U této funkce se počty vrcholů v jednotlivých skupinách lišily opačným způsobem než u předešlé funkce. Skupina 1 měla celkem 111 775 vrcholů. V druhé skupině se nacházelo vrcholů méně, a to 103 878. Stejná tendence však byla pozorována u koeficientu shlukování, kdy v první skupině



Obrázek 43: Porovnání hodnot koeficientu shlukování pro funkci 10- postupný vývoj vrcholů

byla průměrná hodnota vrcholů 0,064 a ve druhé skupině to byla hodnota 0,060. Rozložení hodnot koeficientu shlukování je vidět na obrázku 43.

8.8.3 Účelová funkce č.20

Rozložení počtu vrcholů mezi skupinami zde následovalo stejný trend jako pro funkci číslo 1. V první skupině se nacházelo celkem 203 532 vrcholů. Druhá skupina obsahovala 303 025 vrcholů. Při vzájemném porovnání průměrné hodnoty koeficientu shlukování dosahovala vyšších hodnot skupina jedna.

8.8.4 Účelová funkce č.25

Zde se větší počet vrcholů vyskytoval v první skupině. Konkrétně první skupina obsahovala 150 258 vrcholů. Ve druhé skupině bylo celkem 135 499 vrcholů. U této funkce, na rozdíl od všech tří předchozích, dosahovala vyšší průměrné hodnoty koeficientu shlukování skupina dvě.

9 Závěr

V této práci byly představeny a vysvětleny tzv. biologicky inspirované výpočty. Jedná se o skupinu algoritmů hledající inspiraci v přírodě. Dále došlo k detailnějšímu popisu evolučních algoritmů, které spadají do skupiny biologicky inspirovaných výpočtů. Velký důraz byl kladen také na vysvětlení funkce algoritmu diferenciální evoluce, popis jednotlivých verzí a jejich odlišností.

Další část se věnovala sítím, kde byly představené jednotlivé typy využívaných sítí. V této části byly také detailně popsány vlastnosti počítající se pro sítě, kde některé z nich byly propočítány na modelovém příkladu. Vlastnosti zde byly také zmíněny z důvodu jejich implementace v nástroji, který slouží pro převod běhů algoritmu DE na síť a následnému výpočtu právě těchto vlastností. Tento nástroj byl implementován v rámci této práce.

V rámci této práce byl také implementován program pro detailní zaznamenávání průběhu algoritmu DE, kde vygenerované soubory slouží právě pro převod na síť. Za pomoci tohoto nástroje bylo vygenerovány soubory zaznamenávající 500 běhů algoritmu DE ve verzi DE/rand/1/bin pro každou ze 30 testovacích funkcí. Z těchto 500 běhů bylo pro každou funkci vybráno pouhých 60 běhů, 30 nejlepších a 30 nejhorších.

Z důvodu časové náročnosti, potřebné k analýze jednotlivých skupin běhů, byly pro tuto práci zvoleny 4 testovací funkce, tak aby pokryly všechny kategorie. Běhy algoritmu DE na každé z těchto 4 funkcí byly následně analyzovány za pomocí nástroje pro převod těchto běhů na síť a výpočet vlastností. Analýza byla provedena pro každý z 8 možných převodů. V každém převodu proti sobě byly porovnávány nejlepší a nejhorší běhy na jedné funkci a byly popsány hlavní rozdíly.

Při analýze se ukázalo, že největší rozdíly, u převodů zachycující vzájemné interakce přidáváním a ohodnocováním hran, byly pozorovány u vlastnosti CC. Největší shoda v porovnání jednotlivých vlastností mezi skupinami nastala u převodů, které se zaměřovaly na zachycení negativních interakcí mezi jedinci. U těchto typů převodů bylo na všech zkoumaných funkcích zjištěno stejné rozložení pro hodnoty CC. Skupiny sítí tvořené z nejlepších běhů dosahovaly vyšších hodnot CC oproti skupinám sítí tvořených z běhů nejhorších.

U převodů sledujících postupný vývoj jedinců byla pozorována jedna zajímavost. I přestože v rámci běhů algoritmu DE došlo k menšímu počtu interakcí, kdy výsledkem byl lepší jedinec, našly tyto běhy v celkovém výsledku lepší řešení pro účelovou funkci, než běhy s větším počtem těchto interakcí. K tomuto došlo na testovacích funkcích 1 a 20.

Tyto poznatky by dále mohly být využity při nastavování řídicích parametrů algoritmu DE. Kdy v ideálním případě by za pomoci těchto informací mohl být vytvořen koloběh, kde by za prvé došlo k nastavení výchozích parametrů a spuštění algoritmu DE. Za druhé by byl průběh algoritmu vyhodnocen pomocí převodu na síť a výpočtu vlastností. Za třetí by došlo k zjištění, kde se vyskytuje prostor pro vylepšení, na základě čeho by byly upraveny řídicí parametry algoritmu a znovu spuštěn. Tímto by došlo k uzavření kruhu a řídicí parametry by mohly upravovat běh algoritmu tak, aby dosahoval co nejvyšší účinnosti.

Literatura

[1] Thomas Back. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press, 1996.

[2] Alain Barrat, Marc Barthelemy, and Alessandro Vespignani. The architecture of complex weighted networks: Measurements and models. In *Large Scale Structure And Dynamics Of Complex Networks: From Information Technology to Finance and Natural Science*, pages 67–92. World Scientific, 2007.

[3] S Binitha, S Siva Sathya, et al. A survey of bio inspired optimization algorithms. *International Journal of Soft Computing and Engineering*, 2(2):137–151, 2012.

[4] Stefano Boccaletti, Vito Latora, Yamir Moreno, Martin Chavez, and D-U Hwang. Complex networks: Structure and dynamics. *Physics reports*, 424(4):175–308, 2006.

[5] CESNET. Topologie sítě cesnet2. [online], Dostupné z https://www.cesnet.cz/sluzby/pripojeni/topologie/ , Naposledy navštíveno 24. 4. 2018.

[6] L da F Costa, Francisco A Rodrigues, Gonzalo Travieso, and Paulino Ribeiro Villas Boas. Characterization of complex networks: A survey of measurements. *Advances in physics*, 56(1):167–242, 2007.

[7] Swagatam Das, Sankha Subhra Mullick, and Ponnuthurai N Suganthan. Recent advances in differential evolution–an updated survey. *Swarm and Evolutionary Computation*, 27:1–30, 2016.

[8] Andrea De Montis, Marc Barthélemy, Alessandro Chessa, and Alessandro Vespignani. The structure of interurban traffic: a weighted network analysis. *Environment and Planning B: Planning and Design*, 34(5):905–924, 2007.

[9] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(1):29–41, 1996.

[10] Dario Floreano and Claudio Mattiussi. *Bio-inspired artificial intelligence: theories, methods, and technologies*. MIT press, 2008.

[11] David E Goldberg. *Genetic algorithms*. Pearson Education India, 2006.

[12] DJ Hand. Genetic algorithms in search, optimization and machine learning. *Statistics and Computing*, 4(2):158, 1994.

[13] Karla L Hoffman, Manfred Padberg, and Giovanni Rinaldi. Traveling salesman problem. In *Encyclopedia of operations research and management science*, pages 1573–1578. Springer, 2013.

[14] JUNG. Java universal network/graph framework. [online], Dostupné z http://jung.sourceforge.net, Naposledy navštíveno 18. 4. 2018.

[15] Arpan Kumar Kar. Bio inspired computing–a review of algorithms and scope of applications. *Expert Systems with Applications*, 59:20–32, 2016.

[16] Dervis Karaboga and Bahriye Akay. A survey: algorithms simulating bee swarm intelligence. *Artificial intelligence review*, 31(1-4):61, 2009.

[17] Petr Kovář. Úvod do teorie grafů. *Vysoká škola báňská. Technická univerzita. Ostrava*, 2012.

[18] John R Koza. Genetic programming as a means for programming computers by natural selection. *Statistics and computing*, 4(2):87–112, 1994.

[19] Pavel Krömer. Optimalizace pomocí mravenčích kolonií. *Dostupné z http://homel. vsb. cz/˜ kro080/mravenci. pdf, Naposledy navštíveno 24. 4. 2018*, 2012.

[20] JJ Liang, BY Qu, and PN Suganthan. Problem definitions and evaluation criteria for the cec 2014 special session and competition on single objective real-parameter numerical optimization. *Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore*, 2013.

[21] Sergei Maslov, Kim Sneppen, and Alexei Zaliznyak. Detection of topological patterns in complex networks: correlation profile of the internet. *Physica A: Statistical Mechanics and its Applications*, 333:529–540, 2004.

[22] Mark Newman. *Networks: an introduction*. Oxford university press, 2010.

[23] Mark EJ Newman. The mathematics of networks. *The new palgrave encyclopedia of economics*, 2(2008):1–12, 2008.

[24] Oracle. Java se development kit 8 downloads. [online], Dostupné z http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html, Naposledy navštíveno 18. 4. 2018.

[25] Giuliano Andrea Pagani and Marco Aiello. The power grid as a complex network: a survey. *Physica A: Statistical Mechanics and its Applications*, 392(11):2688–2700, 2013.

[26] Monica Pawlan. What is javafx? [online], Dostupné z https://docs.oracle.com/javafx/2/overview/jfxpub-overview.htm, Naposledy navštíveno 18. 4. 2018.

[27] Stephen Prata. *Mistrovství v C++ 4. aktualizované vydání*. Computer Press, Albatros Media as, 2016.

[28] Kenneth Price, Rainer M Storn, and Jouni A Lampinen. *Differential evolution: a practical approach to global optimization*. Springer Science & Business Media, 2006.

[29] Mikail Rubinov and Olaf Sporns. Complex network measures of brain connectivity: uses and interpretations. *Neuroimage*, 52(3):1059–1069, 2010.

[30] Hans-Paul Paul Schwefel. *Evolution and optimum seeking: the sixth generation*. John Wiley & Sons, Inc., 1993.

[31] John Scott. *Social network analysis*. Sage, 2017.

[32] Yakov Shafranovich. Common format and mime type for comma-separated values (csv) files. 2005.

[33] Lenka Skanderova and Tomas Fabian. Differential evolution dynamics analysis by complex networks. *Soft Computing*, 21(7):1817–1831, 2017.

[34] Rainer Storn. Differential evolution (de). [online], Naposledy navštíveno 13. 11. 2016.

[35] Rainer Storn and Kenneth Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.

[36] Bjarne Stroustrup. *The C++ programming language*. Pearson Education India, 2000.

[37] Lukáš Tomaszek. Dynamika komplexních sítí. 2015.

[38] Peter JM Van Laarhoven and Emile HL Aarts. Simulated annealing. In *Simulated annealing: Theory and applications*, pages 7–15. Springer, 1987.

[39] Ivan Zelinka, Zuzana Oplatková, Pavel Ošmera, Miloš Šeda, and František Včelař. *Evoluční výpočetní techniky*. BEN, 2008.